2013

Dynamic CMOS MIMO Circuits with Feedback Inverter Loop and Pull-Down Bridge

Duo Zhang

Wright State University

DYNAMIC CMOS MIMO CIRCUITS WITH FEEDBACK INVERTER LOOP AND PULL-DOWN BRIDGE

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Engineering

By

Duo Zhang

B.S., *Dalian Institute of Science and Technology, China*, 2011

2013

WRIGHT STATE UNIVERSITY

WRIGHT STATE UNIVERSITY

GRADUATE SCHOOL

July 24, 2013

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER MY SUPERVISION BY Duo Zhang ENTITLED "DYNAMIC CMOS MIMO CIRCUITS WITH FEEDBACK INVERTER LOOP AND PULL-DOWN BRIDGE" BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF Master of Science in Engineering

Chien-In Henry Chen, Ph.D.
Thesis Director

Kefu Xue, Ph.D.
Department Chair

Committee on Final Examination

Chien-In Henry Chen, Ph.D.

Marian K. Kazimierczuk, Ph.D.

Yan Zhuang, Ph.D.

R. William Ayres, Ph.D.
Interim Dean, Graduate School

Abstract

Duo, Zhang. M.S.Egr, Department of Electrical Engineering, Wright State University, 2013. "DYNAMIC CMOS MIMO CIRCUITS WITH FEEDBACK INVERTER LOOP AND PULL-DOWN BRIDGE"

Two novel techniques, feedback inverter loop and pull-down bridge, adopted for multiple-input multiple-output (MIMO) dynamic CMOS circuits have been proposed in this thesis. The pull-down bridge technique optimizes the area and power of a single stage MIMO dynamic CMOS circuits, and the feedback inverter loop (FIL) technique improves the speed of multiple-stage dynamic CMOS circuits. Applying the pull-down bridge to the MIMO dynamic CMOS seven segment decoder, it is shown that common paths of different outputs are shared and optimized, which accounts for 12% speed improvement, 48% power reduction, and 73% area saving, as compared to the conventional logic design. Next, an optimized 64-bit binary comparator implemented by mixed-static-dynamic CMOS with FILs is presented. After partitioning the conventional dynamic CMOS into a mixed-static-dynamic CMOS, optimizing transistor sizes and using the FILs on the critical paths, the proposed design achieves 60% speed improvement and 42% power reduction, as compared to the conventional 64-bit dynamic CMOS comparator.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF TABLES

Acknowledgement

I would like to express the deepest appreciation to my thesis advisor, Professor Henry Chen at Wright State University in Electrical Engineering Department. During my entire research period, Dr. Chen is very patient and shared his vision and knowledge regarding digital design in VLSI and helped me improved my technical skills, my research background and also helped me come out with new innovative ideas and invoked me to reach a higher standard of education. I would also like to thank my thesis committee members Dr. Zhuang Yan and Marian K. Kazimierczuk, who would like to spend their valuable time and serve on my thesis committee.

I would like to thank Xue Hao who shared his views and spent his time enlarging the scope of my work, and Aishwarya Rayipati with whom I discussed my research work and improved the quality of my thesis presentation.

1 INTRODUCTION

1.1 Background

Nowadays, CMOS technology has become a dominant technology for VLSI circuits compared to other digital IC technologies. During the past 20 years, CMOS technology has been continuously scaled down and followed the Moore's law to achieve low cost, high performance, high packing density and low power consumption [1].

In order to attain high-performance applications, MIMO dynamic CMOS logic is widely used in VLSI design. Static CMOS logic is less sensitive to noise, device variations, and has low power consumptions and high stability, but the complementary pull-up network that consists of PMOS transistors is a shortcoming to achieve high speed density [1]. In a dynamic CMOS logic, the pull-up network is replaced by clocked PMOS transistors, and only using the fast pull-down network develops the functionality to reach high-speed and attain high area density. But, dynamic CMOS logic cannot be cascaded directly to form a multiple stage dynamic CMOS circuit. CMOS Domino logic inserts a static inverter between every two dynamic blocks to form domino logic can fix non-cascading problems above, but placing an inverter on the critical path can slow down the speed of circuits. N-P logic (NORA) using two clock phase, cascading a NMOS block with a PMOS block fix the non-invertering issues, but placing PMOS transistors in series to form slow PMOS blocks also suffers from a huge charge-sharing issue. In this thesis, we propose feedback inverter loop adopted for domino CMOS logic, which can speed up the operation of dynamic blocks on the critical and improve the performance of the circuit.

Reducing power consumption is another key concern in VLSI design circuits during recent years. CMOS dynamic logic consumes more power than CMOS static logic. With the technology scaling, the power consumption can be reduced by scaling down the power supply. But, scaling down the power supply can cause performance degraded. Multiple threshold voltage can reduce the power while maintaining speed which requires additional library for different threshold voltages. There are also some techniques to reduce the power consumption like using multiple supply voltage, transistor stacking, adaptive body biasing, and clock gating, etc. In this thesis, we propose a novel technique pull-down bridge for MIMO dynamic CMOS circuits to achieve high packing density and low power consumption.

1.2 Research motivation

Domino CMOS logic is widely used in IC design, when the application requires high-performance. In conventional multiple-stage domino CMOS logic, inserting a static inverter between every two dynamic blocks can slow down the speed of the circuit. Therefore, in this thesis adding a pass transistor (PT) in parallel with inverter to form a feedback inverter loop (FIL) has been proposed to solve this problem. Applying FILs on the critical path of the dynamic circuit, the operation time between every two stage can be reduced and the inverting property still exists. The proposed 64-bit binary comparator is a benchmark circuit to illustrate the performance contribution of employing FILs.

High speed and small area made dynamic CMOS circuits popular for microprocessors, digital signal processors, and even some portable electronic devices. In MIMO dynamic circuits, when the complexity of designing a circuit is increasing, achieving low power consumption and high area density are essential. Applying the proposed pull-down bridge to MIMO dynamic circuits, common paths of different

outputs can be shared, which accounts for reduced area and power consumption. Here, a proposed seven segment decoder is used as a benchmark for application of the pull-down bridge.

1.3 Thesis organization

This thesis is organized as follows: chapter 1 introduces the research background and motivation for dynamic CMOS MIMO circuits with FIL and pull-down bridge. Chapter 2 gives a brief introduction to different applications of CMOS families. Chapter 3 describes the contribution of a novel technique, pull-down bridge used for the MIMO dynamic CMOS seven segment decoder. Chapter 4 proposes an optimized 64-bit binary comparator implemented by mixed static-dynamic CMOS with FILs. Finally, chapter 5 gives the conclusion and future work.

2 CMOS LOGIC FAMILIES

2.1 Digital CMOS IC design category

CMOS stands for Complement Metal Oxide Semiconductor, which is a foremost technology for designing integrated circuit. Usually two types of CMOS are employed for designing digital integrated circuit, static CMOS logic and dynamic CMOS logic. Both styles have their own advantages and disadvantages. According to the design specifications, deciding which logic to implement in designing the circuit is very crucial [2].

2.2 Static CMOS

The most popular logic of CMOS family is Static CMOS logic with complementary PMOS pull-up and NMOS pull-down networks as they are less sensitive to noise and device variations, have low power consumptions and high stability. Fig. 2.1 shows an example of static CMOS "Nand" gate.

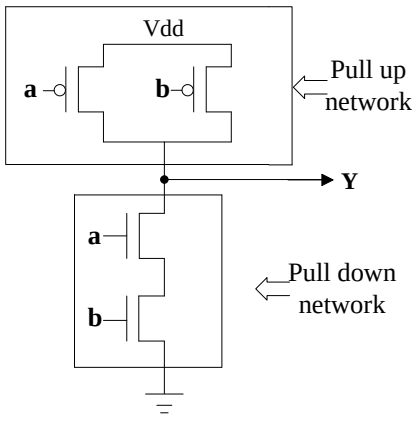

Fig. 2.1 Static CMOS "Nand" gate

2.2.1 Pass transistor logic (PTL)

Pass transistor logic has lower power and smaller area cost compared with

complementary static CMOS [3]. A pass transistor is an NMOS or PMOS transistor controlled by gate terminal behaving like a switch. For an NMOS pass transistor, when the gate terminal voltage is high, it is turned on and the input is passed to the output. But, when the gate terminal voltage is low, there is no connection between the input and output and the output keeps the value of previous stage. A PMOS pass transistor implements the similar way adopted for the NMOS but takes in complemented gate input as a control signal. Pass transistors can be used to implement logic gates and even small circuits. Fig. 2.2 shows an example of designing a PTL "And" gate.

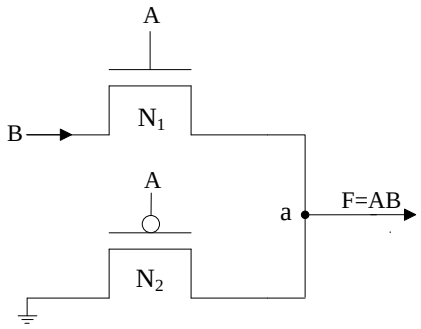

Fig. 2.2 PTL "And" gate

From Fig. 2.2, when both A and B are high, N_1 is turned on. The input B begins to charge the output node 'a' to high. If any of inputs is low, either N_1 or N_2 is turned on. The output node 'a' will be discharged to low. It is obvious that implementing "and" gate using PTL only require 2 transistors as compared to the static complementary CMOS logic using 6 transistors. For IC design using PTL, the area is significantly reduced. On the other hand, the example above uses power inputs instead of power supply for implementing "and" gate, which means it consumes less power. Both of these two advantages can realize large scale integration.

The output of PTL does not have abilities to drive a large load of the next stage circuit is a shortcoming of implementing PTL. Adding buffer at the output can solve

this problem as shown in Fig. 2.3.

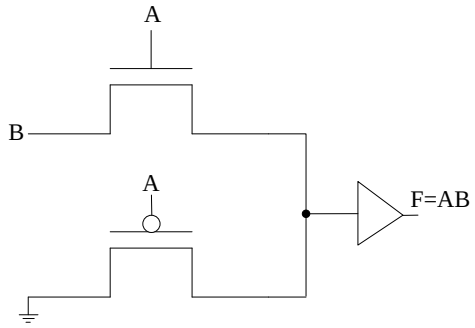

Fig. 2.3 PTL “And” gate with buffer

NMOS transistors pass strong 0 and weak 1 whereas PMOS transistors pass strong 1 and weak 0 which is another drawback in using PTL. The maximum voltage passing to the NMOS transistor cannot go beyond V_{dd}-V_{th} and the minimum voltage passing to the PMOS transistor cannot go below $V_{th.}$ In order to overcome this problem, NMOS and PMOS transistors are parallelly connected and controlled by complementary signal to form a transmission gate as in Fig. 2.4

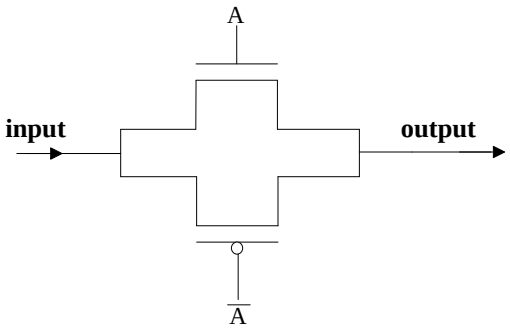

Fig. 2.4 Schematic of Transmission gate

2.3 Dynamic COMS logic

Dynamic CMOS circuit is constructed by two parts, clocked PMOS and NMOS transistors, and an NMOS evaluation block (N-block). Clocked PMOS and NMOS transistors are connected to power supply and ground respectively. An N-block (pull-down network) that generates the function of the circuit is in the middle of the clocked PMOS and NMOS transistors. The dynamic logic is temporary depended on

the output capacitances to hold state. We use a dynamic 2-input “Nand” gate as an example to illustrate how the dynamic CMOS works.

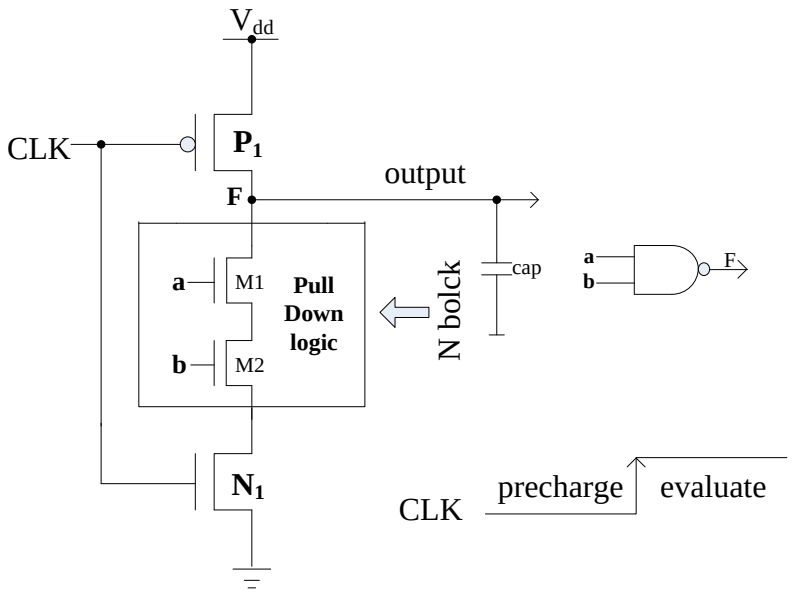

Fig. 2.5 General dynamic CMOS circuit

From Fig. 2.5, when the clock is low, N_1 is off and P_1 is on. V_{dd} charges the output capacitance, so this is called the pre-charge phase. When the clock goes high, N_1 turns on whereas P_1 is turned off and there is no connection between node ‘F’ and V_{dd}. Only when both A and B are high, the output capacitance is discharged to the ground, and this can be called evaluation phase, which generates the function of the circuit.

For dynamic CMOS logic, the number of transistors used is half of the static CMOS logic because the evaluation transistor P_1 replaces the pull-up network. Another advantage is using only NMOS transistors to generate the function of the circuit is much faster than using PMOS transistors, because the mobility of the NMOS electrons is 2-3 times faster than the mobility of the PMOS holes.

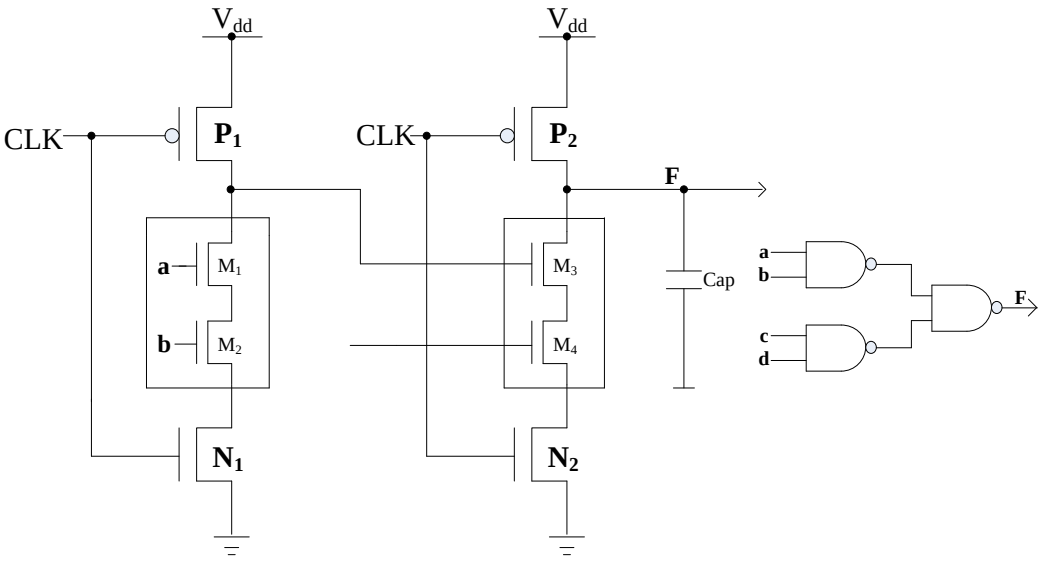

Fig. 2.6 Two-stage dynamic CMOS

Fig 2.6 shows two-stage dynamic CMOS block connected in series. The same clock signal is fed to both dynamic circuits. When the clock is low, all the dynamic blocks are in pre-charge phase and the gate capacitances of the M_3 and M_4 in the second stage are pre-charged to V_{dd}. Next, when the clock goes from low to high at the beginning of the evaluate phase, N_2 and N_1 are turned on and all inputs coming from previous stages are still kept high for a short instance. Therefore, a path to the ground is formed immediately, because M_3, M_4 and N_2 are all on. Then, the cap discharges to ground immediately. This causes the wrong logic at the output of dynamic circuits and can be solved using domino CMOS logic or N-P CMOS logic (NORA).

2.3.1 Domino CMOS logic

Domino CMOS logic, inserting a static inverter between every two dynamic blocks, can cascade multiple stages of dynamic CMOS circuits. When the circuits are in pre-charge phase, all NMOS transistors in N-blocks are turned off. Therefore the outputs of the first stage change first, and then depending on the changes in the first stage, the outputs of the next stage change.

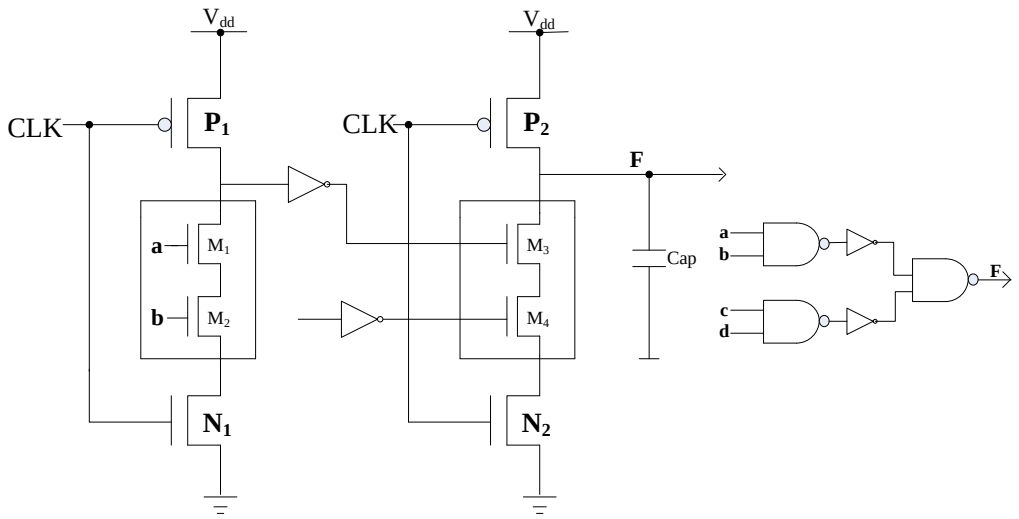

Fig. 2.7 Two-stage domino CMOS logic

Inserting an inverter on the critical path can slow down the speed of the circuit. In order to remove this inverter between two stages, the N-P CMOS logic is introduced below.

2.3.2 N-P CMOS logic

The N-P CMOS logic (NORA) is constructed by cascading an N-block with a P-block [4]. The N-block has been introduced in the previous section and the P-block describes using only PMOS in that block. Fig. 2.8 illustrates how the N-P COMS logic works.

When the global clock is low, all the stages are in the pre-charge phase. The node ‘D’ and ‘F’ are pre-charged to V_{dd} and node ‘E’ is discharged to ground. The outputs of first stage stay at high can turn off the PMOS transistor in the second stage and the outputs of the second stage stay at low can turn off the NOMS transistor in the third stage. In this case, the transistors in both N-blocks and P-blocks are turned off in the pre-charge phase.

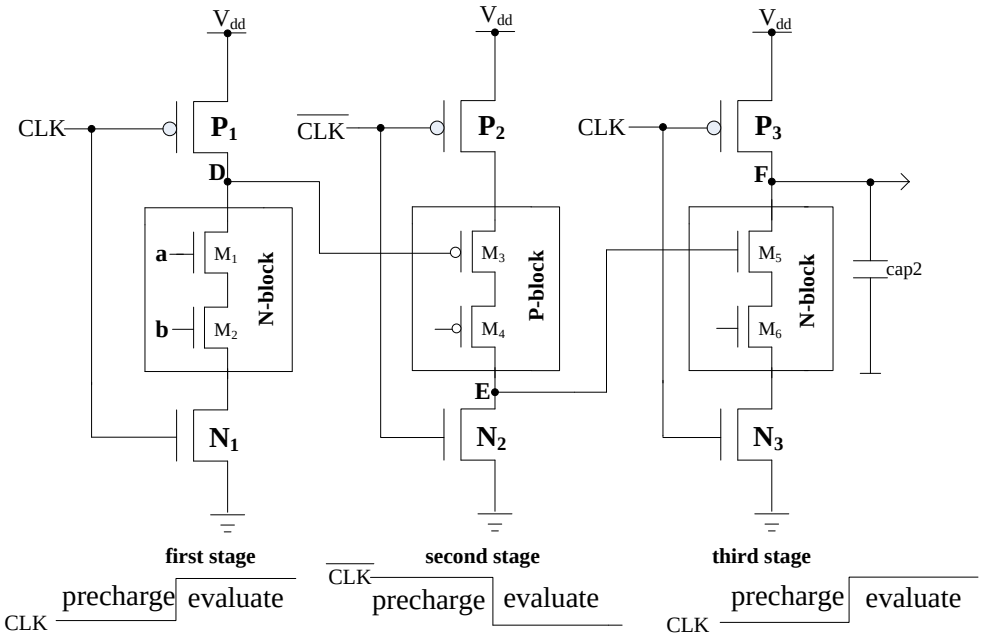

Fig. 2.8 N-P CMOS logic

When the global clock goes high, all the stages are in the evaluation phase. Now, depending on the inputs ‘a’ and ‘b’, the outputs of the first stage can be discharged or stayed high. If all the outputs coming from first stage are low, then the PMOS transistors in the second stage are turned on and output E get charged to high and then can turn on the NMOS transistor in the third stage. In a conclusion, in order to make sure the circuit works properly, all the transistors in the blocks should be turned off at the beginning of the evaluation phase, and then depending on the inputs, the outputs function accordingly.

3 PULL-DOWN-BRIDGE FOR AREA AND POWER SAVING IN MIMO DYNAMIC CMOS

3.1 Introduction of BCD to seven segment decoder (SSD)

BCD to seven segment decoders can be used in applications like digital clocks, electronic meters, and also in some industrial PLC applications to display numerical information [16]. BCD to SSD has four inputs and seven segmented outputs, one output for each LED segment. Seven-segment LED display provides a very convenient way of observing information in the form of numbers from 0 to 9. Fig.3.1 shows an example of the 4-bit BCD input (0100) representing the number 4.

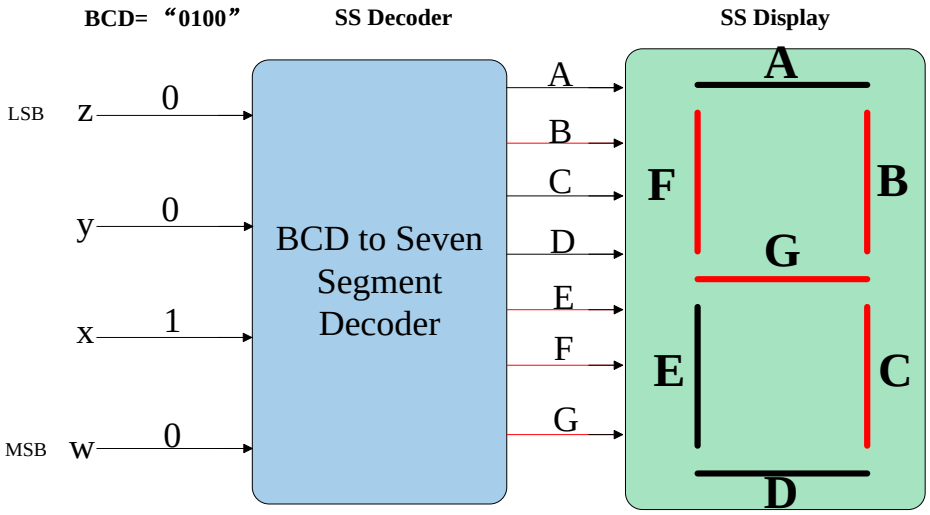

Fig. 3.1 Block diagram of BCD to 7-segment display decoder

Fig.3.1 is constructed using 2 blocks, BCD to SS Decoder, and SS display. BCD to SS Decoder decodes the four-bit BCD input signal to seven segment signals which can be recognized by the SS display.

3.2 SSD using MIMO dynamic CMOS logic

In this section, a seven segment decoder is implemented by using MIMO dynamic CMOS logic instead of conventional design using logic gates. The simulation results show that SSD using MIMO dynamic CMOS logic is 20% faster, 38% power reduction, and 55% transistor saving as compared to the conventional design using logic gates.

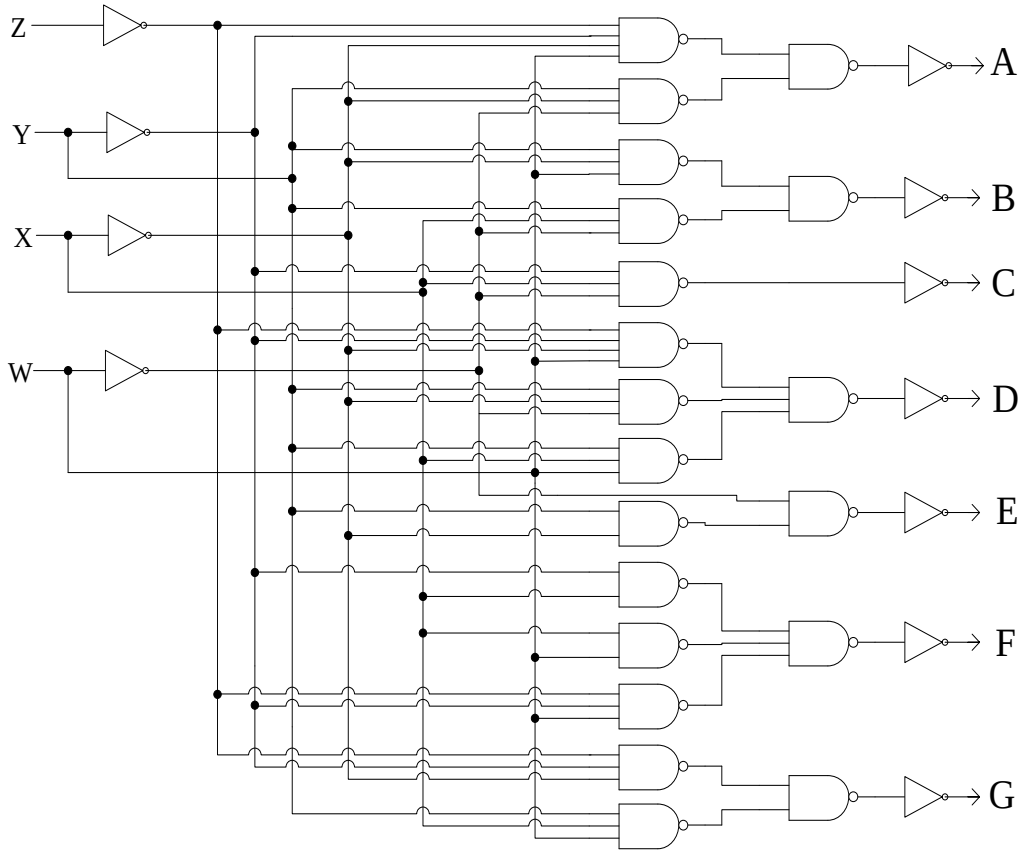

Fig. 3.2 SSD using logic gates

Fig. 3.2 shows the conventional SSD implemented by logic gates. It is evident that large fan-in and fan-out in the design drastically slows down the speed of the circuit and increases the area because of large number of logic gates which in turn increases the power consumption. The truth table of SSD is shown in below

Table: 3.1 Truth table of seven segment decoder

BCD inputs				Decoder Outputs							Display
W	X	Y	Z	A	B	C	D	E	F	G	decimal
0	0	0	0	1	1	1	1	1	1	0	**0**
0	0	0	1	**0**	1	1	**0**	**0**	**0**	0	**1**
0	0	1	0	1	1	**0**	1	1	**0**	1	**2**
0	0	1	1	1	1	1	1	**0**	**0**	1	**3**
0	1	0	0	**0**	1	1	**0**	**0**	1	1	**4**
0	1	0	1	1	0	1	1	**0**	1	1	**5**
0	1	1	0	1	0	1	1	1	1	1	**6**
0	1	1	1	1	1	1	**0**	**0**	**0**	0	**7**
1	0	0	0	1	1	1	1	1	1	1	**8**
1	0	0	1	1	1	1	1	**0**	1	1	**9**
1	0	1	0	X	X	X	X	X	X	X	X
1	0	1	1	X	X	X	X	X	X	X	X
1	1	0	0	X	X	X	X	X	X	X	X
1	1	0	1	X	X	X	X	X	X	X	X
1	1	1	0	X	X	X	X	X	X	X	X
1	1	1	1	X	X	X	X	X	X	X	X

Using MIMO dynamic CMOS logic to implement SSD is introduced below. All the minterms of each output are picked up and placed into the Karnaugh map (K-map) according to their corresponding positions. K-map is a method to simplify Boolean algebra expressions. It groups the common factors and therefore eliminates unwanted variables. Now, we design the circuit using dynamic CMOS, therefore we only need to build the pull-down network. In this case, we group 0's, instead of 1's, according to K-mapping. Fig. 3.3 shows an example of how to group variables in the K-map for output A.

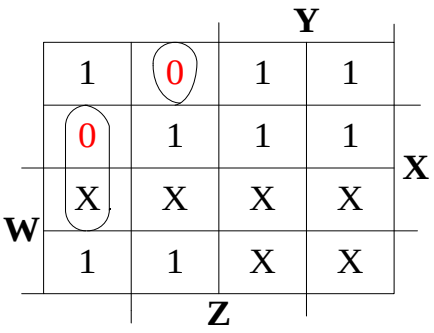

Fig. 3.3 Method of grouping minterms (0's)

The same method applies to all the outputs of SSD, and the simplest expression for each output is developed and listed below.

$$
\begin{aligned}
A &= \bar{w}\cdot\bar{x}\cdot\bar{y}\cdot z + x\cdot\bar{y}\cdot\bar{z}\\
B &= x\cdot\bar{y}\cdot z + x\cdot y\cdot\bar{z}\\
C &= \bar{w}\cdot\bar{x}\cdot y\cdot\bar{z}\\
D &= \bar{w}\cdot\bar{x}\cdot\bar{y}\cdot z + x\cdot\bar{y}\cdot\bar{z} + x\cdot y\cdot\bar{z}\\
E &= \bar{w}\cdot x\cdot\bar{y} + z\\
F &= \bar{w}\cdot\bar{x}\cdot z + \bar{w}\cdot\bar{x}\cdot y + z\cdot y\\
G &= \bar{w}\cdot\bar{x}\cdot\bar{y} + x\cdot y\cdot z
\end{aligned}
\qquad (3.1)
$$

Each term, for example $\bar{w}\cdot\bar{x}\cdot\bar{y}\cdot z$ or $x\cdot\bar{y}\cdot\bar{z}$ in the expression 'A' stands for a pull-down path and each input can be represented by an NMOS transistor. The corresponding inputs decide whether the pull-down path is turned on or off. Fig. 3.4 shows the schematic of SSD using MIMO dynamic CMOS.

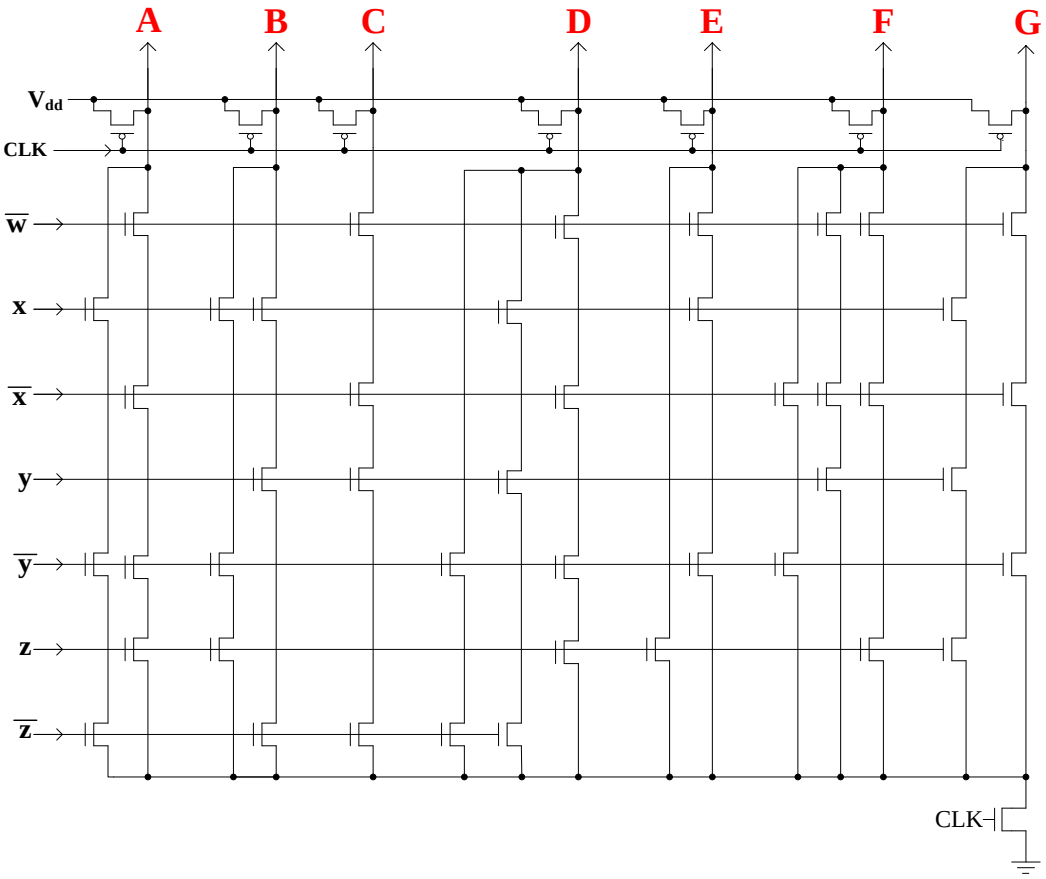

Fig. 3.4 Schematic diagram of MIMO dynamic CMOS SSD

The worst case delay, power consumption, and the number of transistors used in designing an SSD using logic gate vs. MIMO dynamic logic are tabulated below.

Table: 3.2 Performance Comparison of SSD

SSD	Worst delay (ps)	Power (uW)	NO. transistors
Logic gate	530	250	132
Dynamic logic	423	155	60

3.3 Power and area optimization of SSD using Pull-down bridge

We introduce a novel method to optimize the power and area of MIMO dynamic logic using pull-down bridge. The simulation result shows designing MIMO dynamic logic SSD with pull-down bridge achieves additional 20% power reduction and 34%

area saving compared with MIMO dynamic logic SSD.

A. Introduction of Pull-down bridge

Pull-down bridge is used to reduce the power and number of transistors in MIMO dynamic CMOS circuit. Fig. 3.5 shows the basic structure of pull-down bridge. When we observe 0's from the truth table, it is noticed that the output 'A' is a subset of the output 'D', and the output 'D' is a subset of the output 'E'. Thus 'A', 'D', and 'E' are considered as one group. Similarly, the output 'C', subset of the output 'F' is considered as another group.

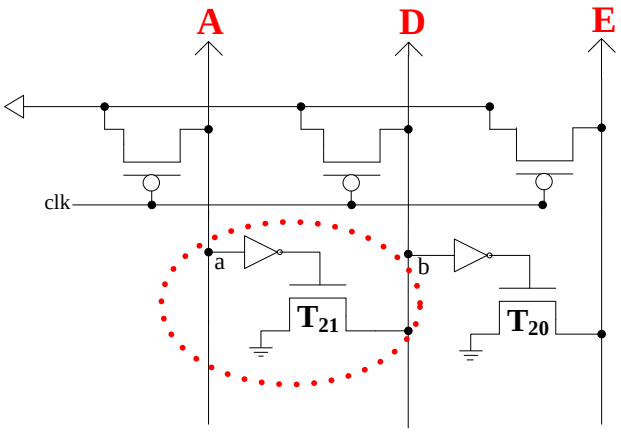

Fig. 3.5 The symbol of pull-down bridge

When the dynamic CMOS circuit enters evaluation phase, if the output 'A' is pulled to ground which implies when node 'A' is '0', transistor T_{21} is turned on and the output 'D' is pulled to ground. Next, the output 'E' is pulled to ground by transistor T_{20}. As a result, the pull-down paths of the output 'A' can be shared with the output 'D', and both paths of 'A' and 'D' can be shared with the output 'E'. Similarly, same method can be applied to other groups too. Thus, the number of transistors and power consumption can be reduced by sharing paths. The new equations for SSD are developed in (3.2) after we choose the subsets and the groups.

$$\begin{cases} A = \bar{y} \cdot (\bar{w} \cdot \bar{x} \cdot z + x \cdot \bar{z}) \\ D = x \cdot y \cdot z \\ E = z \end{cases}$$
$$\begin{cases} C = \bar{w} \cdot \bar{x} \cdot y \cdot \bar{z} \\ F = z \cdot \bar{w} \cdot \bar{x} + z \cdot y \end{cases} \qquad (3.2)$$
$$B = x \cdot \bar{y} \cdot z + x \cdot y \cdot \bar{z}$$
$$G = \bar{w} \cdot \bar{x} \cdot \bar{y} + x \cdot y \cdot z$$

B. Design methodology for MIMO dynamic CMOS with pull-down bridge

The method of sharing transistors in MIMO dynamic CMOS circuit is described as below. According to the new equations, we observed that the combinational term $\bar{w} \cdot \bar{x}$ appears 4 times, $x \cdot y$ appears 3 times, ‘z’ appears 6 times, ‘$\bar{z}$’ 5 times, and ‘$\bar{y}$’ 3 times. From this observation we name the inputs in a horizontal manner from bottom to top starting with the most repeated combinational term. Next, we begin to order the single terms in the similar way. If the same minterm appears in most of the equations, it means that the corresponding transistor can be shared. Therefore, by sharing the transistors, area and power consumption can be reduced. Finally, the order of inputs from bottom to top is $\bar{w}\ \bar{x}\ x\ y\ z\ \bar{z}\ \bar{y}$.

The outputs are arranged from left to right with the smaller subset in the largest group to the left most position. The output E can share paths of outputs D and A through the bridge T_{20} and T_{21}. Similarly, output F shares the path of output C through the bridge T_{19}. Thus, sharing the paths can further reduce the area and power consumption. Finally, the order of the outputs from left to right is A D E C F B G. The schematic of SSD using shared transistors and pull-down bridge is shown in Fig. 3.6.

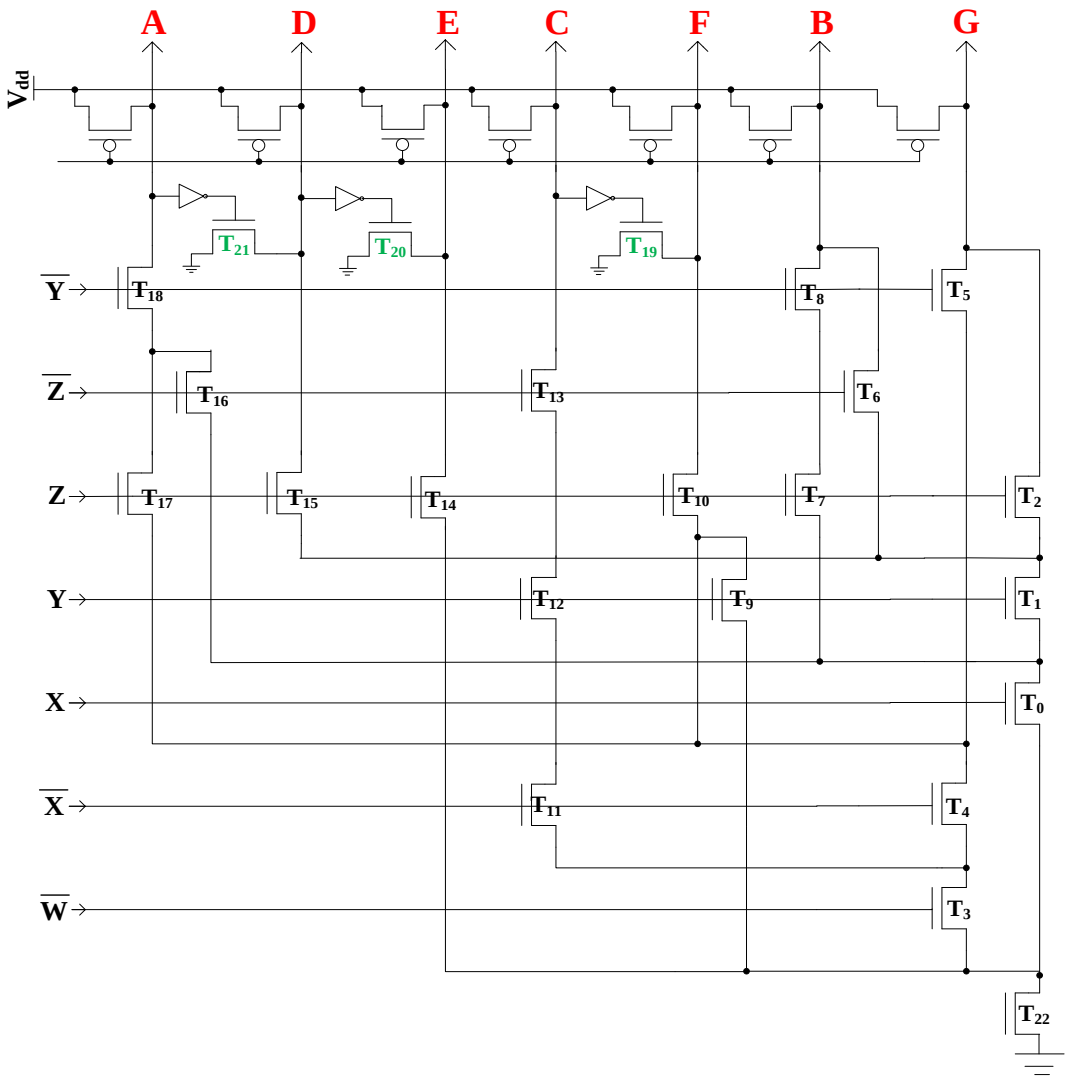

Fig. 3.6 Schematic diagram of novel SSD with pull-down bridge

Table 3.3 shows the performance comparison between the optimized MIMO dynamic CMOS SSD and the conventional SSD using logic gates. It is observed that the novel architecture has 48% less power consumption and requires 72% less area compared to the conventional SSD

Table: 3.3 Performance Comparison of various SSDs

SSD	Worst delay (ps)	Power (uW)	NO. transistors
Logic gates	530	250	132
Dynamic logic	423	155	60
Novel SSD	466	129	36

3.4 Timing optimization of SSD considers transistor size for load balance

The requirement of high-speed operation has become a key factor for integrated circuit. Using CMOS dynamic logic is one of the promising methods for increasing the speed of digital comparator [9-10]. The number and size of the transistors on the critical path decides the performance of CMOS dynamic circuit. In this section, an effective method, named transistor sizing for timing optimization considering load balance of multiple paths is introduced. The basic procedures of using this method to optimize the transistor size are shown in fig. 3.7 [6].

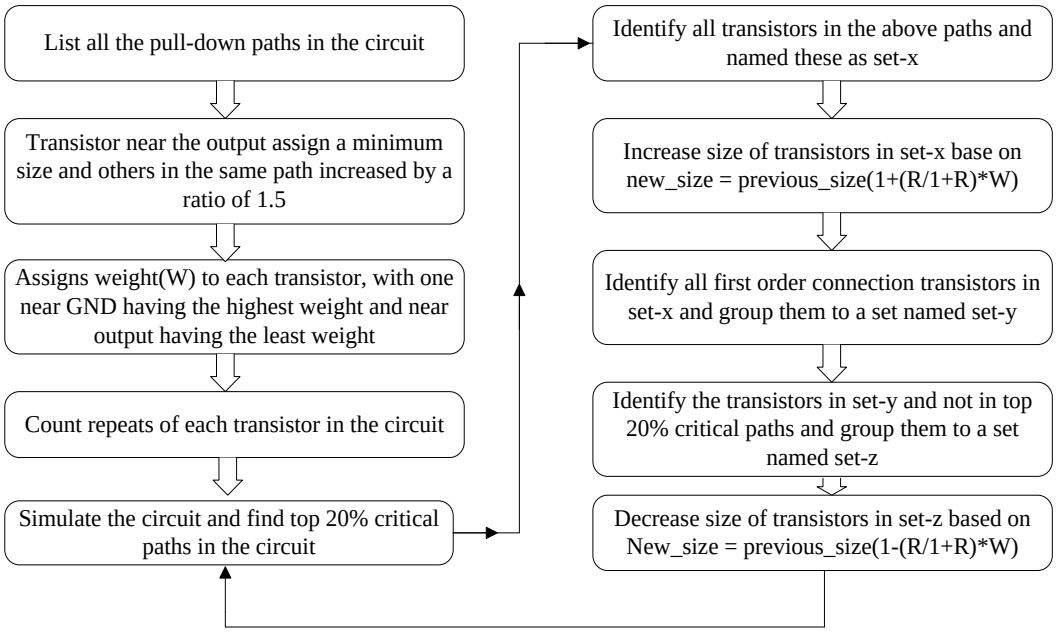

Fig. 3.7 Procedures of transistor size optimization

We optimize the transistor size following the procedures above. First, all the 15 possible paths for SSD are listed in the table 3.4

Table: 3.4 Paths NO. and Transistors for SSD

Path NO.	Transistors	Path NO.	Transistors
1	$T_{18},T_{17},T_4,T_{3,}T_{22}$	9	$T_{13},T_{12},T_{11},T_3,T_{22,}T_{19}$
2	T_{18},T_{16},T_0,T_{22}	10	$T_{10},T_{4,}T_3,T_{22}$
3	$T_{18},T_{17},T_4,T_3,T_{22,}T_{21}$	11	T_{10},T_9,T_{22}
4	$T_{18},T_{16},T_0,T_{22,}T_{21}$	12	T_8,T_7,T_0,T_{22}
5	T_{15},T_1,T_0,T_{22}	13	T_6,T_1,T_0,T_{22}
6	T_{14},T_{22}	14	T_5,T_4,T_3,T_{22}
7	$T_{18},T_{16},T_0,T_{22,}T_{20},T_{21}$	15	T_2,T_1,T_0,T_{22}
8	$T_{13},T_{12},T_{11},T_3,T_{22}$		

Second, we assign weight and count repeats for each transistor with the transistors near the ground having the highest weight, and the repetitions of each transistor that appear in all the 15 paths are shown in table 3.5.

Table: 3.5 Repeats and Weight of transistors in SSD

Repeats	GND						VDD
7			T_0				
6							T_{18}
4	T_3	T_4					
3						T_{16}	T_{21}
2		T_{11}		T_1 T_{12}	T_{17}	T_6 T_{13}	
1				T_9	T_{15} T_{14} T_{10} T_2 T_1		T_{20} T_{19}
Weight	0.5	0.4	0.3	0.2	0.15	0.1	0.05

Then, simulate the circuit following the flow chart in Fig. 3.7 until the final optimized sizes of the transistors are achieved. The final optimized transistor sizes of

SSD are T0 (1380 nm), T1 (360 nm), T2 (360 nm), T3 (2340 nm), T4 (840 nm), T5 (360 nm), T6 (360 nm), T7 (360 nm), T8 (360 nm), T9 (360 nm), T10 (360 nm), T11 (420 nm), T12 (540 nm), T13 (360 nm), T14 (360 nm), T15 (360 nm), T16 (660 nm), T17 (660 nm), T18 (480 nm), T19 (360 nm), T20 (360 nm), T21 (360 nm), and T22 (3520 nm).

The simulation stops when the worst case delay of next iteration cannot be improved. The simulation from minimum size to the final optimized size is listed below. Table 3.6 depicts the rank and delay of top three paths. The transistors in each path are shown in Table 3.4.

Table: 3.6 The rank and delay of top 3 paths

Top 3 paths	Min.size	Ratio 1.5	Iteration1	Iteration 2
1	P_3=466	P_3=307	P_3=271	P_7=262
2	P_1=346	P_7=277	P_7=260	P_3=256
3	P_7=345	P_1=212	P_1=180	P_9=190
Worst-case delay (psec)	466	307	271	262

4 DIGITAL COMPARATOR USING FEEDABCK INVERTER LOOP IN DOMINO COMS

4.1 Feedback Inverter Loop using in CMOS domino logic

CMOS domino logic inserts a static CMOS inverter between every two dynamic CMOS blocks to make sure the entire dynamic circuit works properly. CMOS domino logic has a domino effect when it functions. The output change passes through from the first stage of dynamic CMOS block to the last stage of dynamic CMOS block. Therefore, the inverters inserted between every two stages of dynamic CMOS blocks on the path will substantially increase the path delay. In this case, the delay of critical path is increased, which reduces the speed of the CMOS domino logic. In this chapter, a Feedback Inverter Loop (FIL) is proposed to reduce the path delay as described. The structure of FIL is shown Fig. 4.1, which adds a feedback NMOS pass transistor (PT) controlled by $\overline{\mathrm{CLK}}$.

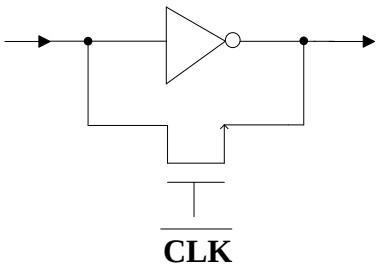

Fig. 4.1 Structure of FIL

The drain and source thermals of the PT are connected to the input and output of the inverter respectively and the gate is controlled by the complemented clock signal.

The global clock signal controls the operation of the circuit. When the clock is low, the dynamic CMOS blocks are in the pre-charge phase. When the clock is high,

the dynamic CMOS blocks go to the evaluate phase. The fact is, in the pre-charge phase the FIL operate and set values, but in the evaluate phase FIL only acts like normal static inverter.

The principle of how the feedback inverter loop works in CMOS dynamic circuit can be illustrated by Fig. 4.2. When the clock is low, dynamic CMOS blocks are in the pre-charge phase and N_3 is turned on. At the same time, P_1, N_3, and N_2 are all switched on. These three transistors form a dynamic loop and they begin to set values at node 'A' and node 'B'. The nodes 'A' and 'B' are charged approximately to $\frac{3}{4}V_{dd}$ and $\frac{1}{4}V_{dd}$ respectively according to the Cadence schematic simulator

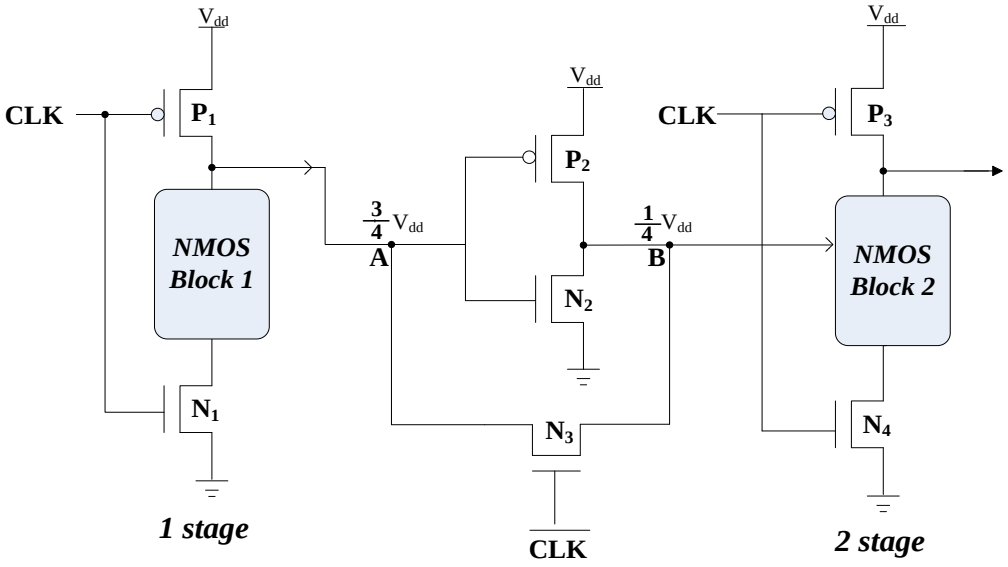

Fig. 4.2 Schematic diagram of FIL in two-stage domino CMOS circuit

There is a significant timing advantage in speed performance after adding a pass transistor N_3 to form a feedback inverter loop in CMOS domino logic. When the dynamic blocks changes from the pre-charge phase to the evaluate phase, the N_3 is turned off. When the NMOS Block 1 is turned on, there is an ON path between ground and node 'A'. Node A begins to discharge to ground from the $\frac{3}{4}V_{dd}$ instead of

the full V_{dd}. In this case, the discharging time of the NMOS block 1 is reduced by 25%. On the other hand, during the discharge phase of node 'A', the P2 will be turned on and N2 will be turned off. V_{dd} will charge the capacitance at node 'B' by P2 and finally turn on the NMOS transistor in the NMOS Block 2. It is obvious that charging the node 'B' to high, starting from $\frac{1}{4}V_{dd}$ is much faster than charging the node B from 0, which in turn will quickly turn on the NMOS in the NMOS Block 2.

The primary feature of static CMOS inverter doesn't change after adding the pass transistor. The $\frac{1}{4}V_{dd}$ at the beginning of evaluate phase won't turn on the NMOS transistors in the NMOS Block 2 and the domino CMOS circuit still retains its function. As a result, the feedback inverter loop improves the timing performance of the Domino CMOS circuit. The application of the feedback inverter loop will be introduced in chapter 4.5.

4.2 Digital comparator

4.2.1 Introduction

A digital comparator, also called magnitude comparator, is a common hardware component in central processing units, microprocessors, and digital signal processing [7]. In digital system, a good compact, low cost, and high performance digital comparator plays an important role in any general-purpose electronics hardware device [8-9]. The function of digital comparator is that it compares the value of two binary inputs, and decides which input is greater, less, equal to the other input [10-11]. Fig. 4.3 is a block diagram of the simple 1-bit digital comparator.

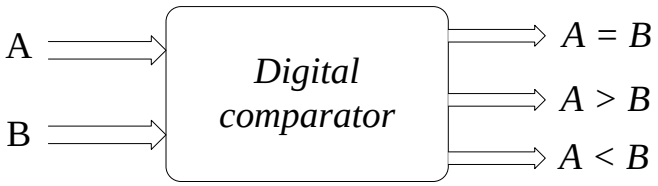

Fig. 4.3 Block diagram of 1-bit digital comparator

The truth table of 1-bit binary comparator is shown in Table 4.1. It illustrates how to compare two single bit digital numbers.

Table: 4.1 Truth table of 1-bit digital comparator

Inputs		outputs		
A	B	A=B	A>B	A<B
0	0	1	0	0
0	1	0	0	1
1	0	0	1	0
1	1	1	0	0

From the truth table, 1-bit digital comparator can be implemented by using logic gates, but if we increase the number of bits, like 8-bit, 32-bit or 64-bit, the complexity of designing the circuit will increase rapidly. On the other hand, huge fan-in and fan-out become another impractical factor for implementing multiple bit digital comparators [11]. In this case, an efficient method in designing a multiple bit digital comparator for high speed and low power is proposed.

4.2.2 Architecture of 64-bit comparator

In this section, a novel single-clock-cycle high-performance priority mixed-dynamic-static CMOS 64-bit binary comparator is introduced and analyzed. Low power and high speed are achieved when implemented by pass transistor (static CMOS) and dynamic CMOS logics. Being simulated by Cadence virtuoso IC 6.1.5 in TSMC 250 nm CMOS technology, the proposed 64-bit comparator is 60% faster, and

42% power reduction than the conventional 64-bit dynamic CMOS comparator.

The tree architecture has been selected to design this parallel 64-bit comparator [12]. The proposed comparator is partitioned into four parallel stages and considered as mixed-static-dynamic CMOS circuit. Fig. 4.4 shows the architecture and block diagram of parallel 64-bit comparator.

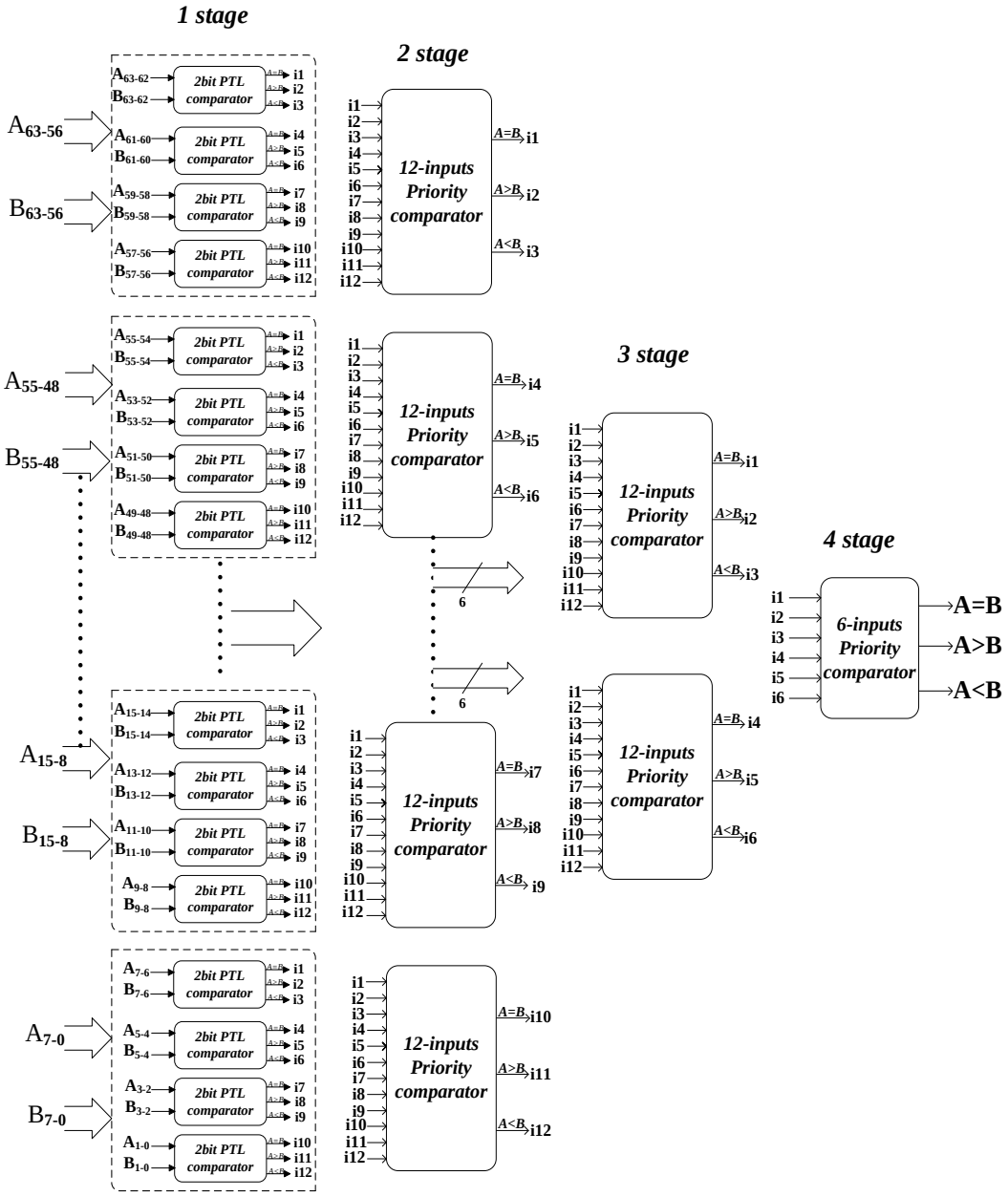

Fig. 4.4 Architecture and block diagram of proposed 64-bit comparator

The first stage of this comparator consists of 32 blocks of 2-bit PTL comparators. The second stage is comprised of 8 blocks of 12-input priority comparator. The third and fourth stages are comprised of 2 blocks of 12-input priority comparators and 1 block of 6-input priority comparator respectively. These four stages are placed in series, and all blocks in each stage are placed in parallel.

A global clock controls the operation of the 64-bit comparator. Figure 4.5 shows the basic timing control of the 64-bit comparator.

Fig. 4.5 Timing control

When the global clock is low, the dynamic CMOS blocks are in the pre-charge phase. All the outputs of dynamic blocks are pre-charged to V_{dd}. At the same time, all the static CMOS blocks begin to operate and prepare the output results for the next dynamic stage. When the global clock goes form low to high, all the dynamic blocks are in the evaluate phase. At this time, the static CMOS has already set the values and gives the results to the input of the dynamic circuit, and then the dynamic circuit begins to evaluate the value and give the final outputs.

The process of how the 64-bit comparator works is discussed as follows. In the first stage, all 32 blocks are implemented by pass transistor logic for low power. In the beginning, all the 128 input signals are fed to the 32 2-bit PTL comparators. These input signals have different priority weight. For instance, the most significant bits of A_{63} and B_{63} have the highest priority weight. After the 32 2-bit PTL comparators compare their inputs bits, 96 internal outputs are generated. Every 2-bit comparator generates 3 binary outputs, A=B, A>B, and A<B, and only one of these three outputs

should be 1. All the internal outputs have their own priority weights. The outputs generated from higher bits have a higher priority weights. Coming to the second stage, the 12 internal outputs generated from the first stage are grouped and fed to the 12-input priority comparator (PC). There are a total of eight 12-input PC in the second stage as there are eight groups coming from the first stage. Next, every 12-input PC will again generate 3 outputs A=B, A>B, and A<B, and a total of 24 binary outputs are generated from the second stage. In the third stage, the same method is applied and only two 12-input PC are needed. Finally, a total of 6 binary outputs are generated, which are fed to the last stage, 6-input PC, to generate the final outputs, AeqB, AgtB, and AltB.

The details of designing and implementation of the 2-bit PTL comparator, the 12-input PC and the 6-input PC will be introduced in the following sections.

4.2.2.1 2-bit Pass transistor logic (PTL) comparator

A low-power 2-bit PTL comparator is used as the basic module for the proposed 64-bit comparator. It is difficult for the output of PTL circuit to drive large loads, hence adding a static CMOS inverter at the output can improve its driving capability. Therefore, the complemented output is observed and implemented, and then an inverter is added to the final output pin to make the PTL work correctly. Table 4.2 is the truth table of the 2-bit comparator.

Table: 4.2 Truth table of 2-bit comparator

Input				Output		
A1	A0	B1	B0	A>B	A=B	A<B
0	0	0	0	0	1	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	0	1
0	1	0	0	1	0	0
0	1	0	1	0	1	0
0	1	1	0	0	0	1
0	1	1	1	0	0	1
1	0	0	0	1	0	0
1	0	0	1	1	0	0
1	0	1	0	0	1	0
1	0	1	1	0	0	1
1	1	0	0	1	0	0
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	0	1	0

K-map is employed to simplify the output equations, and 0's are picked and grouped instead of 1's, according to the above analysis. Three simplest logic expression $\overline{\mathrm{AeqB}}$, $\overline{\mathrm{AgtB}}$, and $\overline{\mathrm{AltB}}$ are derived as shown below.

$$\begin{cases} \overline{\mathrm{AgtB}} = \overline{\mathrm{A0}} \cdot \overline{\mathrm{A1}} + \overline{\mathrm{A0}} \cdot \mathrm{B1} + \overline{\mathrm{A1}} \cdot \mathrm{B0} + \mathrm{B0} \cdot \mathrm{B1} + \overline{\mathrm{A1}} \cdot \mathrm{B1} \\ \overline{\mathrm{AeqB}} = \overline{\mathrm{A1}} \cdot \mathrm{B1} + \mathrm{A1} \cdot \overline{\mathrm{B1}} + \overline{\mathrm{A0}} \cdot \mathrm{B0} + \mathrm{A0} \cdot \overline{\mathrm{B0}} \\ \overline{\mathrm{AltB}} = \mathrm{A1} \cdot \mathrm{A0} + \mathrm{A1} \cdot \overline{\mathrm{B1}} + \overline{\mathrm{A1}} \cdot \mathrm{B0} + \overline{\mathrm{A0}} \cdot \mathrm{B1} + \overline{\mathrm{B0}} \cdot \overline{\mathrm{B1}} \end{cases} \quad (4.1)$$

The equation $\overline{\mathrm{AgtB}}$ is used as an example to illustrate the implementation of a circuit with PTL. A method called decomposition is applied to the logic expression which chooses one variable each time, and decomposes the expression into two sub-expressions. Next, based on those two sub-expressions, we pick another variable and do further decomposition. The same process is repeated, until all the "inputs of the circuit" are achieved. The propagation delay for PTL increases rapidly as the number of stages (numbers of decomposition) increases. Every time we implement a

decomposition process, an additional stage is generated. Therefore, a practical design should have a maximum of four stages by considering the performance of the design. Equation 3.2 shows the basic steps of how to decompose $\overline{AgtB}$.

$$\begin{aligned}\overline{AgtB} &= \overline{a_0}\,\overline{a_1} + \overline{a_0}\,b_1 + \overline{a_1}\,b_0 + b_0 b_1 + \overline{a_1}\,b_1 \\ &= \{a_0 a_1 + \overline{a_0} + \overline{a_1}\,b_0 + b_0 + \overline{a_1}\}_{b_1=1} + \{\overline{a_0}\,\overline{a_1} + \overline{a_1}\,b_0\}_{b_1=0} \\ &= \{(\overline{a_0} + b_0)_{a_1=1} + (1)_{a_1=0}\}_{b_1=1} + \{(0)_{a_1=1} + (\overline{a_0} + b_0)_{a_1=0}\}_{b_1=1}\end{aligned} \tag{4.2}$$

Three-stage decomposition method implements the logic expression of $\overline{AgtB}$. Variable b_1 is picked as a control signal for the third stage and then decomposes the equation based on b_1=1 and b_1=0. Next, variable a_1 is picked for the second stage and continuously decompose the two sub-expressions based on a_1=1 and a_1=0. Finally, variable a_0 is picked for the first stage and is used to decompose the sub-expression $\overline{a_0} + b_0$. Here, $\overline{a_0} + b_0$ is treated as an input for the second stage (Fig.4.6). At last, 1 and b_0 are the inputs for first stage controlled by a_0. $\overline{a_0} + b_0$, 1, and 0 are the inputs for the second stage controlled by a_1. 1 and 0 stand for V_{dd} and ground respectively. Fig. 4.6 shows the schematic diagram of AgtB

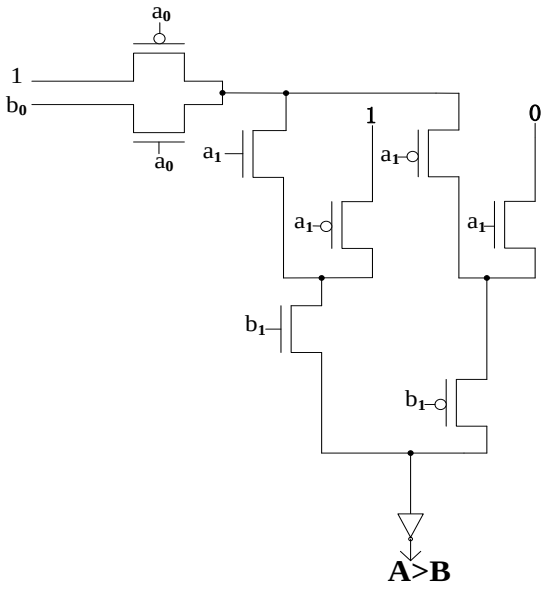

Fig. 4.6 Schematic diagram of AgtB

The same decomposition process are implemented to the equations $\overline{\text{AeqB}}$ and $\overline{\text{AltB}}$ (4.1). Fig 4.7 shows the schematic diagram of 2-bit PTL comparator.

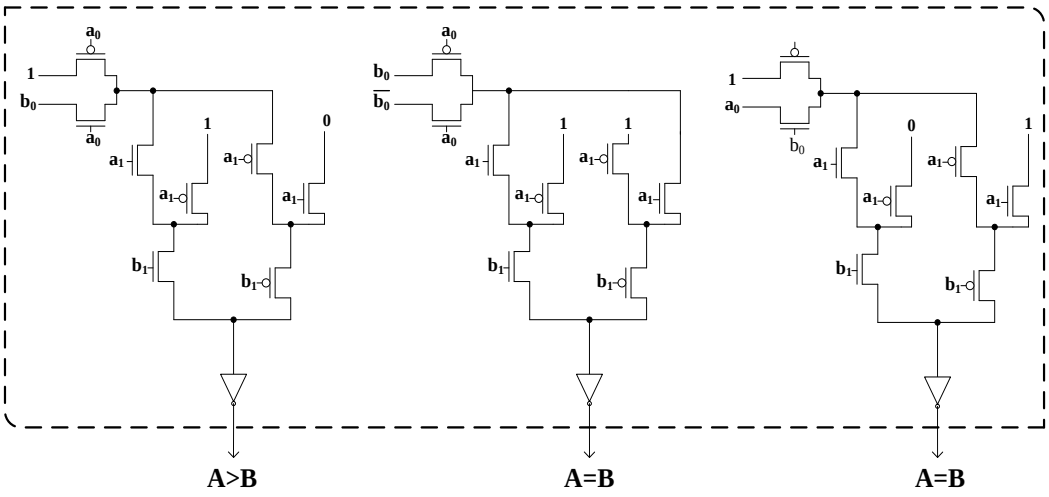

Fig. 4.7 Schematic diagram of 2-bit PTL comparator

Table 4.3 shows the performance comparison among different 2-bit CMOS comparators. It is evident that using PTL to implement 2-bit comparator has obvious advantages.

Table: 4.3 Comparison of different 2-bit CMOS comparators

CMOS style	Power (uW)	Worst delay(ps)	NO. of Trans
Dynamic CMOS	219	210	38
Static CMOS	144	567	64
PTL	106	331	38

4.2.2.2 6-input and 12-input priority comparators

The priority encoder (PE) is a basic electric component in the digital system [13]. It allocates priority levels to every input. The output corresponds to the input which has the highest priority. After the input with the highest priority is identified, all other inputs with the lower priority will be ignored [14]. Priority comparator (PC) has different levels of priority inputs. The idea of designing the priority encoder can be applied to design the priority comparator.

The 6-input and 12-input priority comparators (PCs) are introduced in this section. These two PCs are designed as modules to use in the proposed 64-bit comparator. There is one 6-input PC and ten 12-input PCs in the 64-bit comparator.

A. 12-input priority comparator

12-input PC works with four 2-bit comparators to form the proposed 8-bit comparator. Four 2-bit comparators work as the first stage of 8-bit comparator and the 12-input PC works as a second stage. The 12 outputs generated from the four 2-bit comparators are fed to 12-input PC according to their corresponding positions. These 12 outputs have their own priority weighting. In fact, the 2-bit comparators compare and generate the results and the 12-input PC classifies these results to export the final compared outputs. According to the outputs generated from the four 2-bit comparators, we can get the truth table of 12-input PC shown in Table 4.4. The weights of all the

outputs from the first stage are also listed in Table 4.5.

Table: 4.4 Truth table of 12-input PC

Inputs												Outputs		
i_1	i_2	i_3	i_4	i_5	i_6	i_7	i_8	i_9	i_{10}	i_{11}	i_{12}	AeqB	AgtB	AltB
1	0	0	x	x	x	x	x	x	x	x	x	1	0	0
0	1	0	x	x	x	x	x	x	x	x	x	0	1	0
0	0	1	1	0	0	x	x	x	x	x	x	1	0	0
0	0	1	0	1	0	x	x	x	x	x	x	0	1	0
0	0	1	0	0	1	1	0	0	x	x	x	1	0	0
0	0	1	0	0	1	0	1	0	x	x	x	0	1	0
0	0	1	0	0	1	0	0	1	1	0	0	1	0	0
0	0	1	0	0	1	0	0	1	0	1	0	0	1	0
0	0	1	0	0	1	0	0	1	0	0	1	0	0	1

Table: 4.5 Weights of 12 inputs

2-bit comp	A_{7-6} B_{7-6}	A_{5-4} B_{5-4}	A_{3-2} B_{3-2}	A_{1-0} B_{1-0}
inputs	AeqB = i_1 AgtB = i_2 AltB = i_3	AeqB = i_4 AgtB = i_5 AltB = i_6	AeqB = i_7 AgtB = i_8 AltB = i_9	AeqB = i_{10} AgtB = i_{11} AltB = i_{12}
Weight	4	3	2	1

The logic equations are listed based on the Table 4.4 and Table 4.5

$$\begin{cases} \mathrm{AeqB} = \mathrm{i1} \cdot \mathrm{i4} \cdot \mathrm{i7} \cdot \mathrm{i10} \\ \mathrm{AgtB} = \mathrm{i2} + \mathrm{i1} \cdot \mathrm{i5} + \mathrm{i1} \cdot \mathrm{i4} \cdot \mathrm{i8} + \mathrm{i1} \cdot \mathrm{i4} \cdot \mathrm{i7} \cdot \mathrm{i11} \\ \mathrm{AltB} = \mathrm{i3} + \mathrm{i1} \cdot \mathrm{i6} + \mathrm{i1} \cdot \mathrm{i4} \cdot \mathrm{i9} + 1\mathrm{i} \cdot \mathrm{i4} \cdot \mathrm{i7} \cdot \mathrm{i12} \end{cases} \qquad (4.3)$$

The 12-input PC is implemented using MIMO dynamic CMOS logic, according to the logic equations. Fig. 4.8 shows the schematic diagram of the 12-input PC

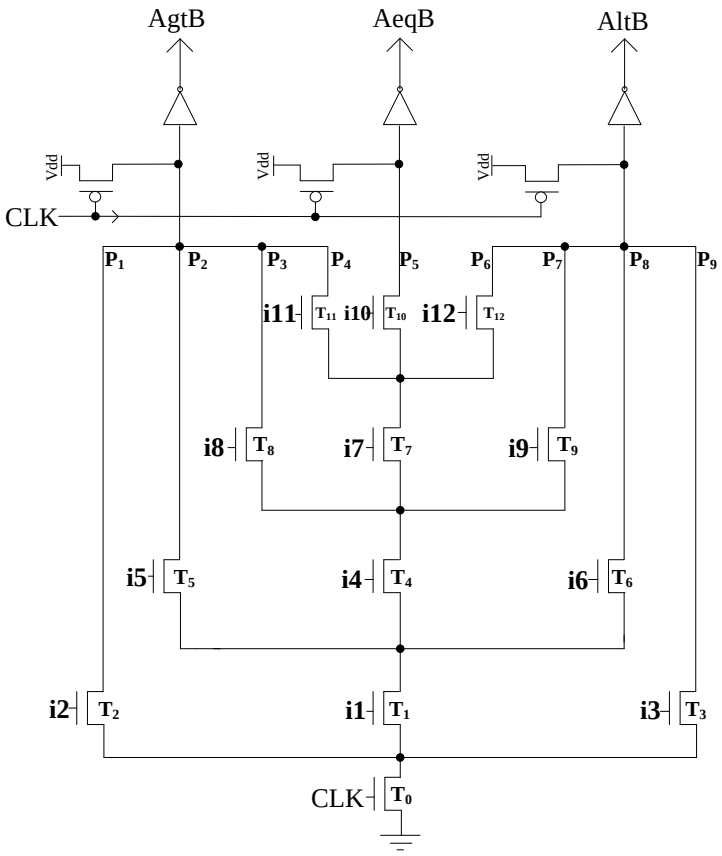

Fig. 4.8 Schematic diagram of 12-input PC

From the Fig. 4.8, the architecture of AgtB and AltB are exactly same and symmetrical. It is evident that increasing the number of transistors on a path will increase the discharging time of that path and therefore increase the output pull-down path delay [6]. P_1 and P_9 have only two transistors on their paths, so they are the best case for output AgtB=1 or AltB=1. Whereas P_4 and P_6 have five transistors on their paths, so that is the worst case for AgtB=1 or AltB=1. The output AeqB=1 only has one path P_5 which has five transistors. There are three paths P_4, P_5, and P_6 having five transistors, but the critical path for the 12-input PC are P_4 and P_6. These two paths have 3 fanouts individually. Transistors T_1, T_2, andT_3 are channel-connected with T_{11} on P_4; and T_9, T_6, and T_3 are channel-connected with T_{12} on P_4. These channel-connected transistors are treated as a capacitive load for T_{12} and T_{13} which

results in an increased delay on P_4 and P_6 as compared to P_5 with T_{10} which has an extra capacitive load on the drain side.

B. 6-input priority comparator

The idea of implementing and analyzing the 6-input PC is as same as the 12-input PC. 6-input The PC can be considered as the second stage of 4-bit comparator and 64-bit comparator with two 2-bit PTL comparators and two 32-bit comparators in the first stage. The truth table is shown in Table 4.6.

Table: 4.6 Truth table of 6-input PC

Inputs						Outputs		
i_1	i_2	i_3	i_4	i_5	i_6	AeqB	AgtB	AltB
0	0	1	x	x	x	0	0	1
0	1	0	x	x	x	0	1	0
1	0	0	0	0	1	0	0	1
1	0	0	0	1	0	0	1	0
1	0	0	1	0	0	1	0	0

From the truth table the logic expressions of 6-input PC are listed below

$$\begin{cases} \mathrm{AeqB} = \mathrm{i1} \\ \mathrm{AgtB} = \mathrm{i2} + \mathrm{i1} \cdot \mathrm{i5} \\ \mathrm{AltB} = \mathrm{i3} + \mathrm{i1} \cdot \mathrm{i6} \end{cases} \qquad (4.4)$$

CMOS dynamic logic is used to implement the above logic expression. The schematic diagram of the 6-input PC is shown in Fig.4.9

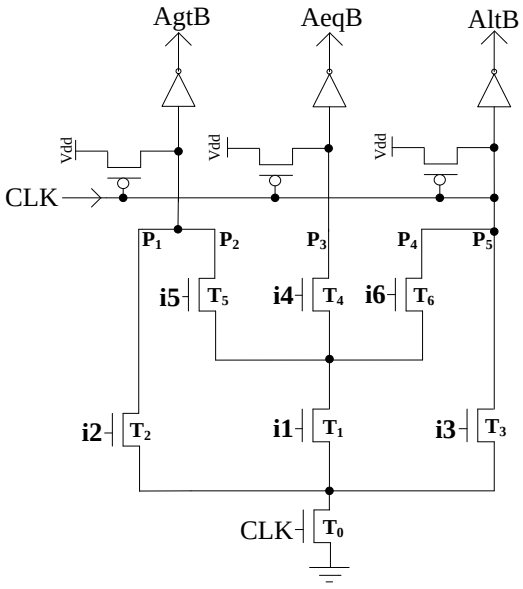

Fig. 4.9 Schematic diagram of 6-input PC

4.2.2.3 8-bit comparator

In this section, the 8-bit comparator implemented by four 2-bit PTL comparators and one 12-input priority comparator is introduced and analyzed. This comparator can also be used as a basic block for the proposed 64-bit comparator. Implementing the circuit by flattening the logic expression of the 8-bit comparator is not a practical way because of highly complex truth table and logic expressions. Hence tree structure can be applied to achieve higher bit digital comparators. Fig 4.10 shows the architecture of the 8-bit comparator.

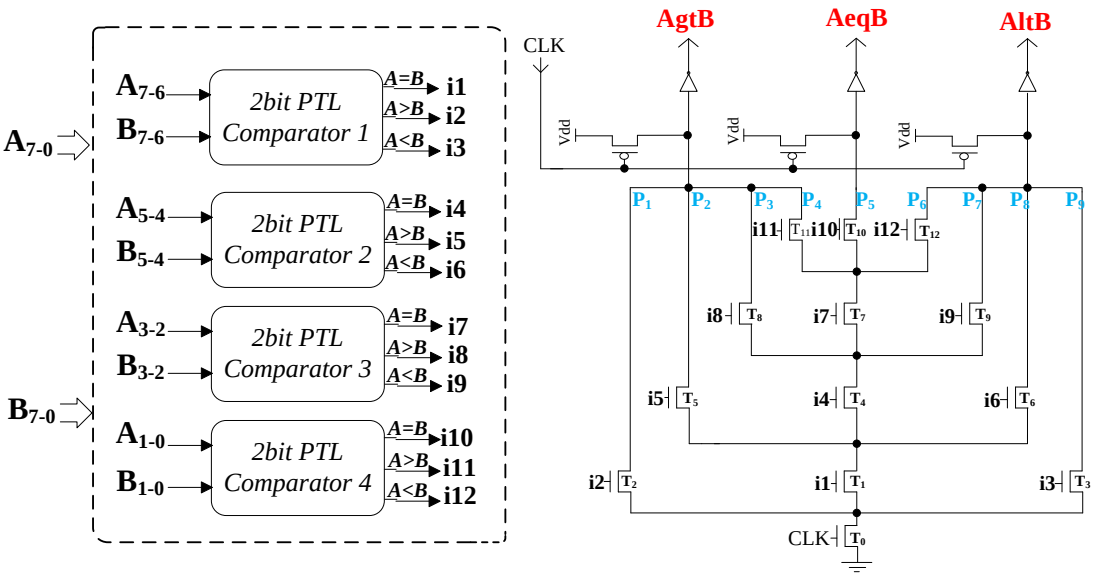

Fig. 4.10 Architecture of 8-bit comparator

The 8-bit comparator is implemented by mixed-static-dynamic CMOS logic. In the pre-charge phase, the 12-input dynamic PC is pre-charged to V_{dd}. Simultaneously, the four 2-bit PTL comparators compare the initial inputs. The outputs coming from the 2-bit PTL comparators have priority weights and are ready for the dynamic CMOS to evaluate. When the circuit enters the evaluate phase, the 12-input priority comparator begins to operate and the 8-bit PC immediately recognizes the inputs with higher priority weight. There are total 9 pull-down paths in 12-input PC, and only one of them is turned on during each evaluate phase. If any one of P_1, P_2, P_3, and P_4 are turned on, then the output AgtB=1. If any one of P_6, P_7, P_8, and P_9 are turned on, output AltB=1. The output AeqB only happens when P_5 is turned on.

4.2.2.4 32-bit and 64-bit comparators

In this section, the idea of designing 8-bit comparator is extended to the multi-bit comparator. The 32-bit comparator is considered as a basic module of the proposed 64-bit comparator and is implemented by using four 8-bit comparators and one 12-input PC. The operation of 32-bit comparator is similar to the operation of the

8-bit comparator. Four 8-bit comparators is used as basic modules in the first stage and their outputs are fed to the 12-input PC. Fig. 4.11 shows the block diagram of 32-bit comparator.

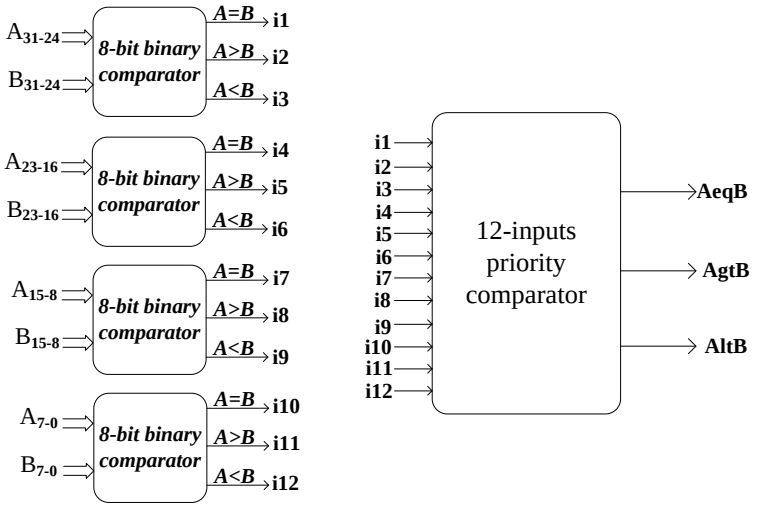

Fig. 4.11 Block diagram of 32-bit comparator

The 64-bit comparator is implemented by using two 32-bit comparators and one 6-input PC. The block diagram of 64-bit comparator is showed in Fig.4.12

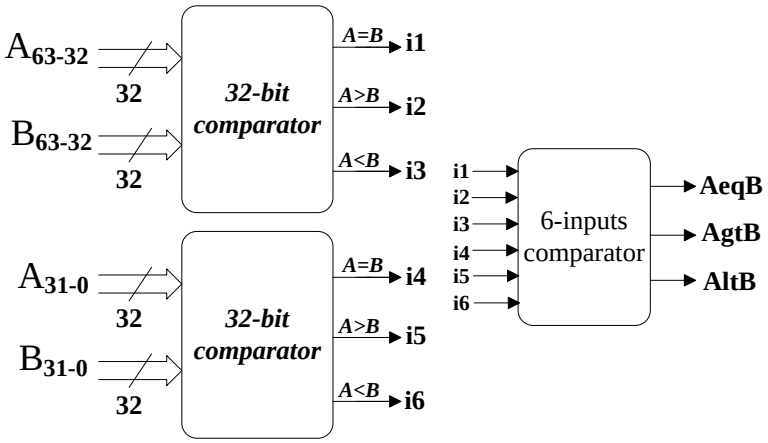

Fig. 4.12 Block diagram of 64-bit comparator

4.2.3 Clock tree design for 64-bit binary comparator

In ultra-deep submicron IC design there are millions of blocks that need clock as a control signal. One global clock-control apparently cannot handle such a huge load. For synchronous system, each block requires same rise and fall time to make sure that

the circuits work correctly. Clock signals are very sensitive, and worse clock signal can cause large clock skew and jitter, which consequently affects the circuit behavior. Therefore, clock tree design is very essential in designing a high performance clock distribution network.

For the proposed 64-bit comparator, there are totally 11 dynamic blocks. Every dynamic block has three clocked PMOS and one clocked NMOS and the gate terminal of these clocked transistors are connected to the clock signal. The size of clocked PMOS and NMOS transistors are 780nm and 4600nm. These large size transistors have huge gate capacitance which acts as a load for the clock. The performance of a global clock driving huge loads is shown in Fig. 4.13. As a result, it will affect the performance of the circuit.

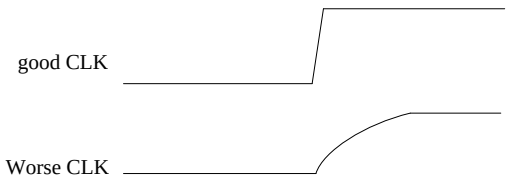

Fig. 4.13 Clock signal

Clock inverters are used typically to implement the clock tree design. For the proposed clock tree, a ratio of 1.5 is used to enlarge the size of both PMOS and NMOS in the inverter design for every stage. Fig. 4.14 shows the structure of proposed clock tree

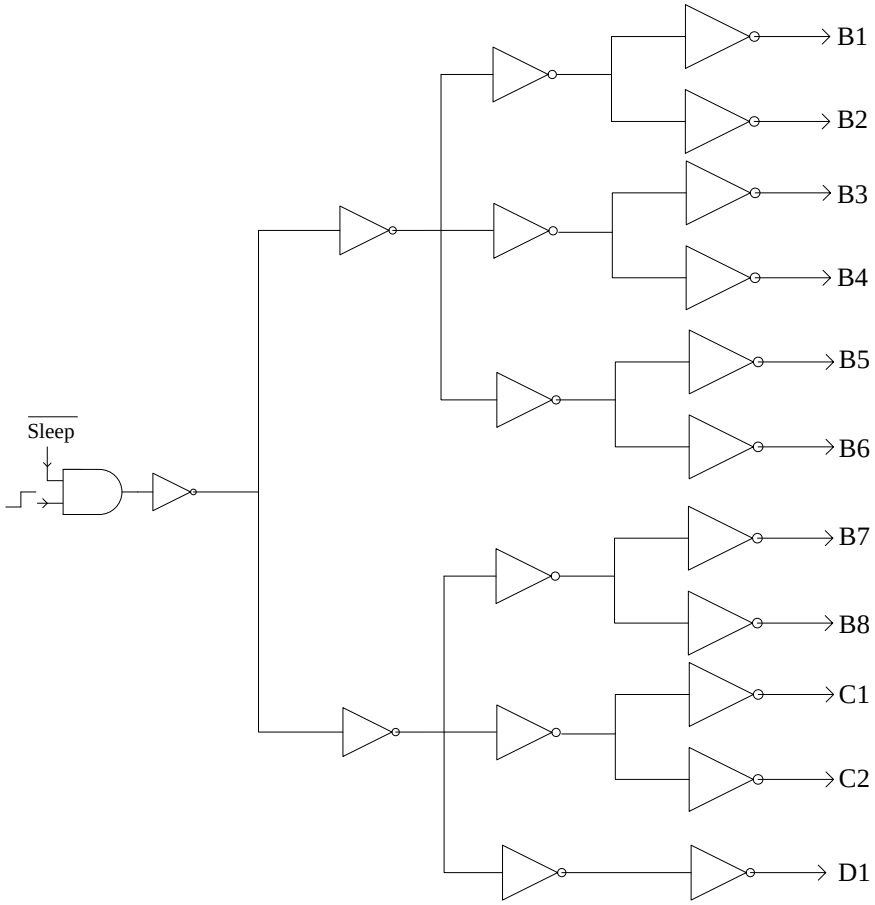

Fig. 4.14 Architecture of proposed clock tree

A control unit implemented by AND gate is added for this clock tree design. When the circuit wants to stop, the sleep mode is triggered to '1' else to '0'. The outputs $B_{1\text{-}8}$, $C_{1\text{-}2}$, and D_1 are fed to the eight 12-input PC in the second stage, two 12-input PC in the third stage and one 6-input PC in the last stage of proposed 64-bit comparator.

4.3 Timing optimization for 12-input and 6-input PC consider transistor size for load balance

For the proposed 64-bit comparator, the dynamic propagation delay is measured in the evaluation phase, from the point where the second stage begins to operate until the final outputs are generated from the last stage. For all the three stages using

dynamic CMOS logic, the 12-input and 6-input priority comparators are used as the basic modules both in parallel and series. In this case, improving the speed of these modules can make a significant contribution to increase the speed of the 64-bit comparator. In this section, the same method of transistor size optimization is implemented and the simulation results show that the optimized 12-input priority comparator is 64% faster than using the minimum sized 12-input priority comparator. The optimized 6-input priority comparator also has 41% speed improvement.

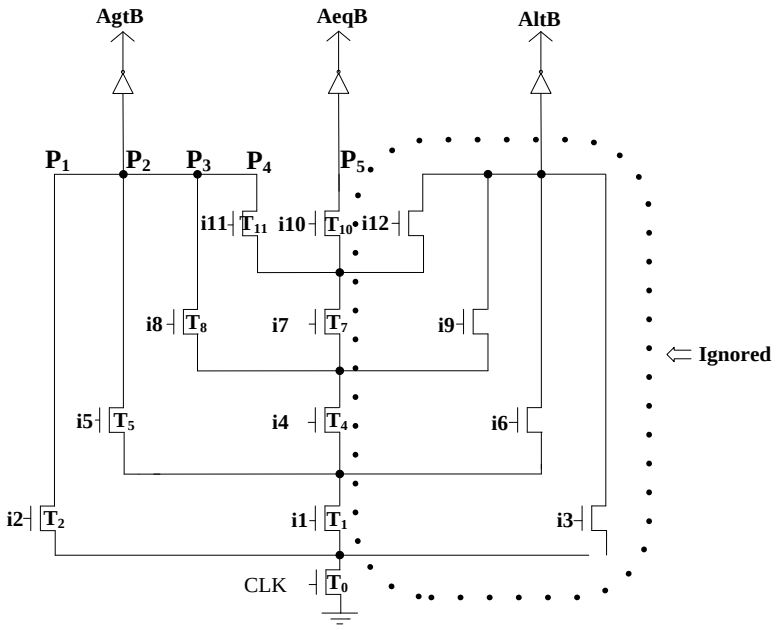

Fig. 4.15 Timing path for 12-input PC

This section presents transistor size optimization for 12-input PC. The schematic diagram of the 12-input PC is shown in Fig.4.15. It is obvious that the schematic diagram of AgtB and AltB are symmetrical. Transistor sizes of all the 4 paths belonging to AgtB are same as those of AltB. During optimization, the paths belonging to AltB are not considered, as shown in Fig.4.15, but the load of AltB still exists at the nodes on the critical path. In this case, only optimizing the transistor size belonging to the AgtB and AeqB will reduce the complexity of optimization process.

Table 4.7 shows the five timing paths and Table 4.8 presents the weight and repeats of transistors from Fig. 4.15

Table: 4.7 Timing path of the 12-input PC

Path	Transistors
1	T_2, T_0
2	T_5, T_1, T_0
3	T_8, T_4, T_1, T_0
4	T_{11}, T_7, T_4, T_1, T_0
5	T_{10}, T_7, T_4, T_1, T_0

Table: 4.8 Weight and repeats of transistors in 12-input PC

Repeat	GND			VDD
7	T_1			
5		T_4		
3			T_7	
1				T_2, T_5, T_8 T_{11} T_{10}
Weight	0.5	0.4	0,3	0.2

Table 4.9 is summarized after five iterations of the proposed MIMO dynamic CMOS circuit, the final transistor sizes of 12-input priority comparator are T_2 (360 nm), T_3 (360 nm), T_5 (360 nm), T_6 (360 nm), T_8 (480 nm), T_9 (480 nm), T_{11} (660 nm), T_{12} (660 nm), T_{10} (360 nm), T_7 (1470 nm), T_4 (3400 nm), T_1 (6400 nm), and T_0 (9700 nm).

Table: 4.9 Worst case delay of each iteration

Path rank	Min. size	Ratio 1.5	Iteration I	Iteration II	Iteration III	Iteration IV	Iteration V
1	P_4	P_4	P_4	P_4	P_4	P_4	P_4
2	P_5	P_5	P_3	P_5	P_3	P_3	P_2
3	P_3	P_3	P_5	P_3	P_5	P_5	P_3
4	P_2	P_2	P_2	P_2	P_2	P_2	P_5
5	P_1	P_1	P_1	P_1	P_1	P_1	P_1
Worst.del (Psec)	327	210	186	172	152	135	117

The architecture of 6-input PC is similar to 12-input PC. The schematic of 6-input PC is shown in Fig. 4.8 which is optimized by the same method adopted for the 12-input PC. The final optimized sizes of the transistors are as follows T_2 (360 nm), T_3 (360 nm), T_5 (540 nm), T_6 (540 nm), T_4 (360 nm), T_1 (1200 nm), and T_0 (1800 nm).

4.4 Prevent charge sharing by using additional pre-charge PMOS

In CMOS domino logic, charge sharing is an undesirable phenomenon. It occurs when the charge stored at the output capacitance is shared with other transistor junction capacitances in the evaluation phase. Charge sharing can degrade the output voltage level or even cause erroneous output response.

In the 12-input PC design, as the transistor sizes increase, charge sharing becomes a non-ignorable issue. The conventional way to deal with this problem is to add a weak PMOS called ‘keeper’ at the output to keep the dynamic node high during evaluation phase if it is not being pulled down through N-block. Fig.4.16 shows the structure of keeper used in CMOS domino logic

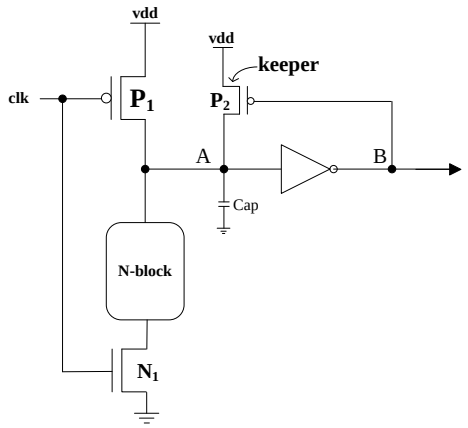

Fig. 4.16 Structure of keeper

Using ‘keeper’ can degrade the performance of the circuit, assuming that there is a pull-down path in the N-block. At the beginning of evaluation phase, node A begins

to discharge, but the voltage drop at node A is not enough to turn the P_2 off. Here, even if node A begins to discharge, V_{dd} can still charge the node A through the transistor P_2. This increases the power consumption and the discharge time.

The additional pre-charge PMOS transistor (PPT) that is proposed can not only fix the charge sharing, but also decrease the power consumption and the discharge time as compared to the conventional PMOS keeper. Fig. 4.17 shows the schematic diagram of 12-input PC with additional pre-charge PMOS.

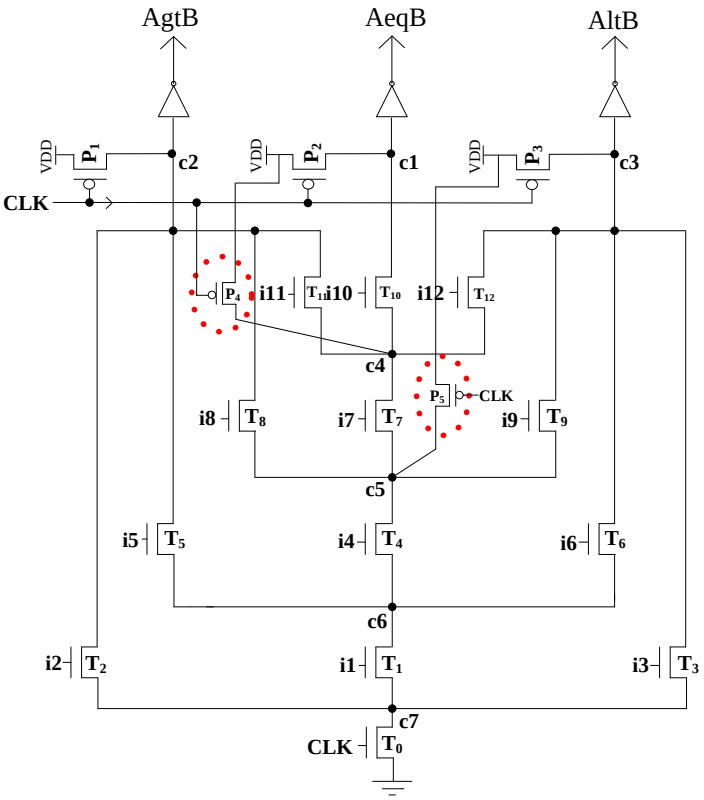

Fig. 4.17 Schematic of 12-input PC with prechage PMOS

The worst-case charge sharing problem experienced with 12-input PC can be described as follows. During the first clock period N in the evaluation phase, T_{10}, T_7, T_4, T_1, and T_0 are all turned on. The nodes c_1, c_4, c_5, c_6, and c_7 are discharged to the ground, and outputs are AeqB=1, AgtB=0, and AltB=0. During the N+1 pre-charge phase, all inputs are low, assuming that they come from the previous domino logic

blocks. If N+1 is in the evaluation phase, T_{10}, T_7, and T_4 goes high, while T_1 still stays low. The node c_1 should be kept high and output AeqB should be 0, according to the functionality of the 12-input PC. But, when T_{10}, T_7, and T_4 are turned on, these transistors form a discharging path connecting node c_1 to the node c_6. There is no charge in the parasitic capacitance of transistors at node c_6, c_5, and c_4, because these nodes are discharged to ground during the previous clock cycle N. Increasing the transistor size will increase its parasitic capacitance. The total parasitic capacitance on path consisted with T_{10}, T_7, T_4, T_1, and T_0 is greater than the parasitic capacitance at node c_1. This causes a charge sharing between node c_1 and c_6 that causes the voltage stored at c_1 to discharge all the parasitic capacitances of the transistors on that path. As a result, the voltage drop at c_1 can cause the output error.

This charge sharing problem can be eliminated by adding pre-charge PMOS transistors P_4 and P_5 on the pull down network at those positions that have large parasitic capacitance. These additional pre-charge PMOS transistors are controlled by the clock signal. During N+1 clock cycle in the pre-charge phase, all transistors in N-network are turned off except P_4 and P_5. These two on transistors, P_4 and P_5 begin to pre-charge the internal nodes c_4 and c_5. When the circuit enters the evaluation phase both P_4 and P_5 are turned off by the clock. Thus, c_4 and c_5 will have enough charges that won't let c_1 discharge and cause the output error.

The use of additional pre-charge PMOS transistors depends on two factors, noise tolerance and timing requirement, which are mutually affected. Adding a pre-charge PMOS at every node in the circuit will increase the noise tolerance, but the timing performance is affected because every node on the discharge path must discharge from its maximum charge. Increasing the discharge time will increase the delay of the circuit. Table 4.10 depicts the performance analysis of 12-input PC using different

number of PMOS transistors.

Table: 4.10 Performance analysis of EPPTs in 12-input PC

Case NO.	Worst. Delay(ps)	Power(uw)	Min.V_{dd} (V)
Case 1	230	205	1.60
Case 2	198	188	1.80
Case 3	224	200	1.75
Case 4	242	212	1.70

Case1: using keeper

Case2: using one PPT on node c_4

Case 3: using two PPTs on node c_5 and c_6

Case 4: using three PPTs on the node c_5, c_6 and c_7 respectively.

4.5 Timing optimization of 64-bit binary comparator using FILs

FILs applied to the 64-bit comparator results in further timing optimization, based on the primary timing optimization using the optimized transistor size. FILs operate between every two dynamic blocks that are in series on the critical path. It is observed from the simulations that using FILs on the critical path can achieve another 18% delay improvement based on the optimized transistor size.

The FILs are only inserted on the critical path, because inserting FILs between all the dynamic blocks can cause extra power consumption. The critical path for proposed 64-bit comparator is achieved when the input vectors are $A_{63\text{-}0}$ = 00,..,00 and $B_{63\text{-}0}$ = 00,..,01 or $A_{63\text{-}0}$ = 00,..,01 and $B_{63\text{-}0}$ = 00,..,00. In this case, the FILs will be used on every A=B output of 12-input PCs. FILs cannot be inserted in the last stage, because stable results, full V_{dd} or ground are desired at the final outputs. Fig.4.16 shows the block diagram of 64-bit comparator with FILs on the critical path.

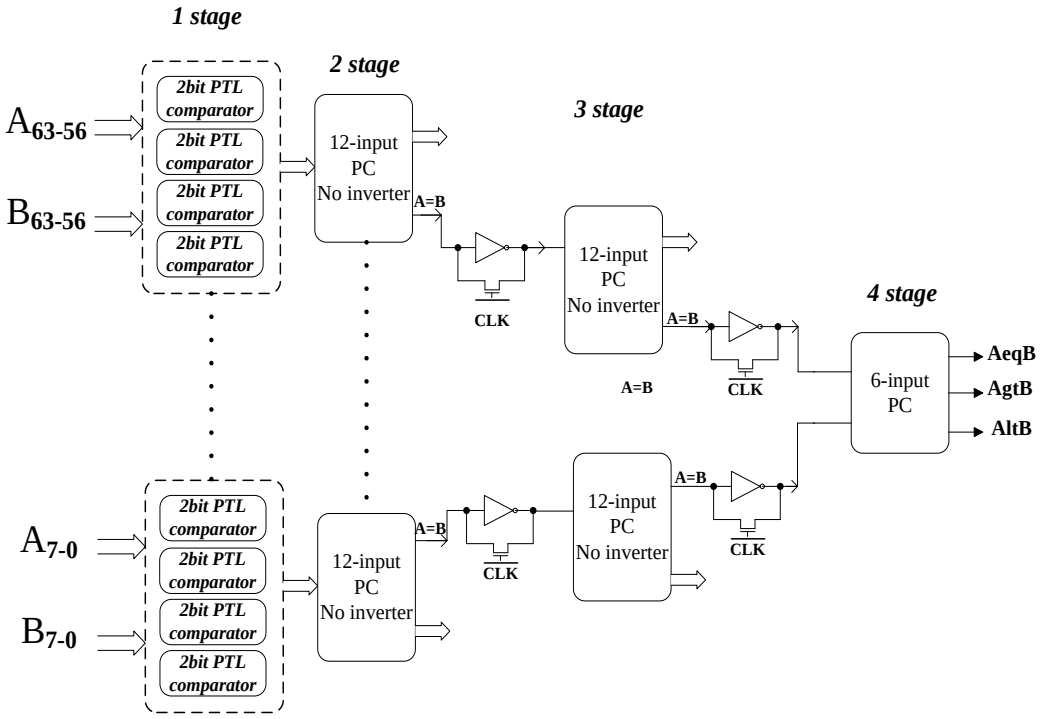

Fig. 4.18 Block diagram of 64-bit comparator with FILs

The inverters at the output in every 12-input PC and a pass transistor are shunted to form FILs. Table 4.11 shows the performance comparison of mixed-static-dynamic optimized size 64-bit comparator with FILs and 64-bit comparator without FILs.

Table: 4.11 Performance comparison of 64-bit various comparators

64-bit comp	Worst. Delay (ps)	Max. freq (GHz)	Power (uW)
Mixed-Opt_size	410	1.1	3.13
Mixed-Opt_size With FILs	355	1.25	4.81

4.6 Performance analysis for 64-bit comparator

In this section, we analyze the timing, power, and throughput of proposed 64-bit comparator and compare the performance of this design with the previous design.

A. Performance analysis for proposed 64-bit comparator

Mixed-static-dynamic CMOS logic can reduce the power consumption and improve the throughput of 64-bit comparator as compared to conventional dynamic CMOS 64-bit comparator. Transistor size optimization and feedback inverter loop can remarkably increase the speed of 64-bit comparator.

Fig. 4.19 depicts the timing analysis of the proposed 64-bit comparator. The worst case delay of 3-stage dynamic blocks is 355ps, and the static block is 325 ps. The total delay of this design is recorded as 680ps. The time taken to operate static block is not considered and only the dynamic delay can be considered as the delay of the entire circuit is 355ps.

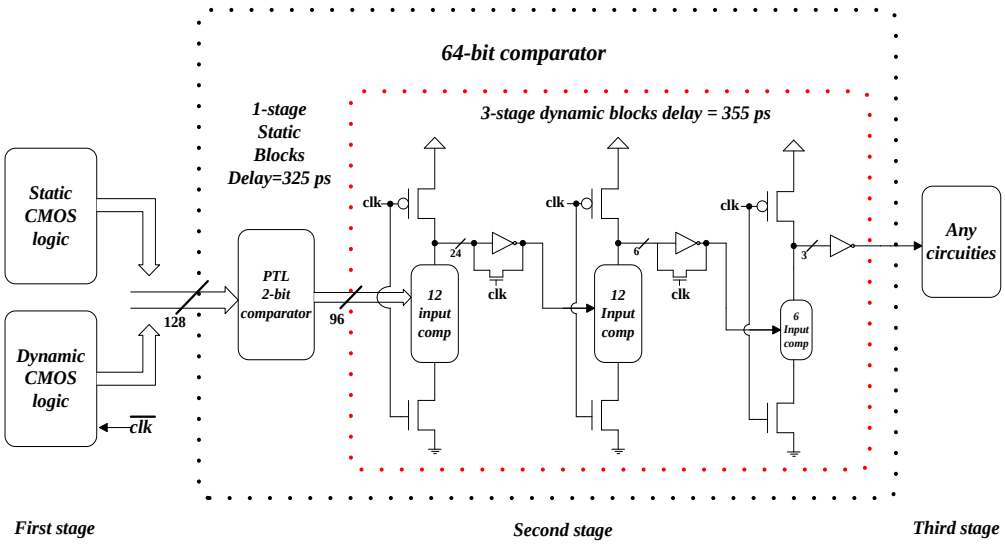

Fig. 4.19 Timing analysis of the proposed 64-bit comparator

From Fig. 4.19, the outputs coming from first stage must arrive at the second stage to meet the setup time requirement. If the first stage uses dynamic CMOS logic, the global clock is complemented whereas the 64-bit comparator uses normal clock. Under such circumstances, all stages can work correctly. Table 4.12 shows the performance of various 64-bit comparators at different levels of optimization.

Table: 4.12 Performance of various 64-bit comparators in this thesis

Style of comparator	Worst Delay(ps)	Max. freq (GHz)	Clock Period (ns)	Worst Power (mW)	2ns Power (mW)	Min. V_{dd}
All dynamic Min-size	882	0.5	2	8.35	8.35	1.8
Mixed logic Min-size	704	0.55	1.8	2.27	2.02	1.8
Mixed logic Opt-size	410	1.11	0.9	5.92	3.13	2.1
Mixed logic Opt-size+FIL	**355**	**1.25**	**0.8**	**7.87**	**4.81**	**2.3**

Denote: (1) The power consumption for all the 64-bit comparator listed above includes the power of clock tree.
(2) The clock period stands for the minimum clock period that the comparators function properly.
(3) The '2ns power' stands for the power consumption that is measured when clock period= 2ns.
(4) Min. V_{dd} stands for the minimum V_{dd} that the comparators function properly.

B. This work compare to the prior works [14]

Table 4.13 summarizes the delay, power, and transistors used for various 64-bit comparators from prior works. All the simulated results are achieved at schematic level in the Cadence design environment. The proposed 64-bit comparator (250 nm) performs faster and uses less number of transistors than that of previous work (180 nm).

Table: 4.13 Performance comparison of various 64-bit comparators

Publication	This work	Chuang and Li [14]	Huang and Wang [10]	Lam and Tsui [9]	Kim and Yoo [15]
CMOS(nm)	250	180	180	180	180
Delay (ps)	355	642	752	453	1005
power (uW)	5800	1224	1364	3102	2194
NO. Transistors	1485	2988	2382	5519	2469

5 CONCLUSION AND FUTURE WORK

5.1 Conclusion

This thesis introduces: 1) the pull-down bridge technique to reduce the area and power consumption of a single stage MIMO dynamic CMOS logic, and 2) the feedback inverter loop increases the speed of multi-stage domino CMOS logic.

Employing pull-down bridge for conventional MIMO dynamic CMOS SSD, the common path for different outputs are shared and it is observed that the proposed SSD achieves 48% power reduction and 72% area saving as compared to the conventional SSD using logic gates. Using the feedback inverter loop between every two dynamic CMOS blocks on the critical path further optimizes the speed of 64-bit comparator. This is analyzed from the final results, which shows 60% speed improvement and 42% power saving than the conventional 64-bit dynamic CMOS comparator. All the circuits are implemented and simulated by TSMC 250 technology in Cadence 6.1.15 design environment with 2.5V power supply.

5.2 Future work

The feedback inverter loop in Fig. 4.2, when it works in the pre-charge phase, there is a short current path between V_{dd} and ground formed by the on-transistors P_1, N_3, and N_2, which will increase the power consumption. A technique to minimize and rescue this power consumption is under investigation.

6 REFERENCE

[1] James B. Kuo and Jea-Hong Lou, “Low-Voltage CMOS VLSI Circuit”, pp.1, and pp.163-166. 1999

[2] K. Yelamarthi, “Process Variation-aware Timing Optimization with Load Balance of Multiple Paths in Dynamic and Mix-static-dynamic CMOS Logic,” PhD Dissertation, June 2008.

[3] S.-F. Hsiao, M.-Y. Tsai, and C.-S. Wen, “Transistor Sizing and Layout Merging of Basic Cell in Pass Transistor Logic Cell Library,” *IEEE International Symposium on*. pp.89-92, April.2008

[4] N. F. Goncalves and H. J. De Man, “NORA: A race-free dynamic CMOS technology for pipelined logic structures,” *IEEE J. Solid-State Circuits,* vol. 18, pp. 261-266, June 1983

[5] http://www.electronics-tutorials.ws/combination/comb_6.html

[6] K. Yelamarthi and C-I. H. Chen, “Transistor Sizing for Load Balance of Multiple Paths in Dynamic COMS for Timing Optimization,” *Proceedings of IEEE/ACM International Symposium on Quality Electronic Design*, pp. 426-431, Mar 2007

[7] G. Sharma, U. Nirmal, and Y. Misra, “A Low Power 8-bit Magnitude Comparator with Small Transistor Count using Hybrid PTL/CMOS Logic,” *IJCEM International Journal of Computational Engineering & Management*, vol. pp.110-115,12, April 2011.

[8] H.-M. Lam and C.-Y. Tsui, “High-performance single clock cycle CMOS comparator,” *Electron.Lett.*, vol. 42, no.2, pp. 75-77, Jan 2006

[9] H.-M. Lam and C.-Y. Tsui, “A MUX-based high-performance single cycle CMOS comparator,” *IEEE Trans. Circuits Syst. II, Exp. Briefs,* vol. 54, no.7, pp. 591-595, Jul.2007.

[10] C.-H. Huang and J.-S. Wang, “High-performance and power-efficient CMOS comparators,” *IEEE J. Solid-State Circuits,* vol. 38, no. 2, pp. 254-262, Feb. 2003

[11] C.-C. Wang, C.-F. Wu, and K.-C Tsai, “1-GHz 64-b high-speed comparator using ANT dynamic logic with two-phase clocking,” *Proc. Inst. Elect. Eng. Comput. Digital Techn.,* vol. 145, no. 6, pp.433-436,Nov.1988.

[12] C.-W. Ku, L.-G. Chen, T.-D. Chiueh, and H.-M. Jong, “Tree-Structure Architecture and VLSI Implementation for Vector Quantization Algorithms,” *IEEE International Symposium on,* vol.6, pp.139-142. Jan.1994.

[13] J.-S. Wang and C.-H. Huang, “High-speed and low-power CMOS priority encoders,” *IEEE J. Solid-State Circuit,* vol. 35, pp. 1511-1514, Oct.2000.

[14] Pierce Chuang, David Li, and Manoj Sachdev, Fellow, IEEE, “A Low-Power High-Performance Single-Cycle Tree- Based 64-Bit Binary Comparator”, *IEEE Transactions on Circuits and Systems-II: Express Briefs*, Vol.59, No. 2, February 2012

[15] J.-Y. Kim and H.-J. Yoo, “Bitwise competition logic for compact digital comparator,” *in Proc. IEEE Asian ASSCC*, pp. 59-62, 2007

[16] http://en.wikipedia.org/wiki/Seven-segment_display